33rd International Symposium on Graph Drawing and Network Visualization

GD 2025, September 24–26, 2025, Norrköping, Sweden

Edited by

Vida Dujmović

Fabrizio Montecchiani



LIPIcs – Vol. 357 – GD 2025 www.dagstuhl.de/lipics

Editors

Vida Dujmović
University of Ottawa, Canada
vida.dujmovic@uottawa.ca

Fabrizio Montecchiani
University of Perugia, Italy
fabrizio.montecchiani@unipg.it

ACM Classification 2012
Mathematics of computing → Graph theory; Mathematics of computing → Graph algorithms; Human-centered computing → Graph drawings

ISBN 978-3-95977-403-1

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at https://www.dagstuhl.de/dagpub/978-3-95977-403-1.

Publication date
November, 2025

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

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

The copyright is retained by the corresponding authors.

Digital Object Identifier: 10.4230/LIPIcs.GD.2025.0

LIPIcs – Leibniz International Proceedings in Informatics

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

Editorial Board

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

ISSN 1868-8969

https://www.dagstuhl.de/lipics

Contents

Preface
Vida Dujmović and Fabrizio Montecchiani 0:ix–0:x

Sponsors
..... 0:xi

Authors
..... 0:xiii–0:xvii

Invited Talks

A Sketch of Parameterized Complexity
Hans L. Bodlaender 1:1–1:1

Transforming Graph Visualization through AI and Human-AI Collaboration
Huamin Qu 2:1–2:1

Regular Papers

On the Structure of Normalized Models of Circular-Arc Graphs I. Hsu's Approach
Tomasz Krawczyk 3:1–3:15

Heuristics for Exact 1-Planarity Testing
Simon D. Fink, Miriam Münch, Matthias Pfretzschner, and Ignaz Rutter 4:1–4:19

Constrained Flips in Plane Spanning Trees
Oswin Aichholzer, Joseph Dorfer, and Birgit Vogtenhuber 5:1–5:18

Approximating Barnette's Conjecture
Michael A. Bekos, Michael Kaufmann, and Maximilian Pfister 6:1–6:7

Same Quality Metrics, Different Graph Drawings
Simon van Wageningen, Tamara Mchedlidze, and Alexandru C. Telea 7:1–7:13

Tangling and Untangling Trees on Point-Sets
Giuseppe Di Battista, Giuseppe Liotta, Maurizio Patrignani, Antonios Symvonis, and Ioannis G. Tollis 8:1–8:18

The Page Number of Monotone Directed Acyclic Outerplanar Graphs Is Four or Five
Jawaherul Md. Alam, Michael A. Bekos, Martin Gronemann, and Michael Kaufmann 9:1–9:17

The Bend Number of Cocomparability Graphs
Todor Antić, Vit Jelínek, Martin Pergel, Felix Schröder, Peter Stumpf, and Pavel Valtr 10:1–10:23

Towards a Better Understanding of Graph Perception in Immersive Environments
Lin Zhang, Yao Wang, Ying Zhang, Wilhelm Kerle-Malcharek, Karsten Klein, Falk Schreiber, and Andreas Bulling 11:1–11:19

33rd International Symposium on Graph Drawing and Network Visualization (GD 2025).
Editors: Vida Dujmović and Fabrizio Montecchiani
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Flipping Odd Matchings in Geometric and Combinatorial Settings
Oswin Aichholzer, Sofia Brenner, Joseph Dorfer, Hung P. Hoang, Daniel Perz, Christian Rieck, and Francesco Verciani 12:1–12:18

Separability of Witness Gabriel Drawings
Carolina Haase, Philipp Kindermann, William Lenhart, and Giuseppe Liotta 13:1–13:18

OOPS: Optimized One-Planarity Solver via SAT
Sergey Pupyrev 14:1–14:19

Crossing Number of Simple 3-Plane Drawings
Miriam Goetze, Michael Hoffmann, Ignaz Rutter, and Torsten Ueckerdt 15:1–15:19

Structural Parameterizations of k-Planarity
Tatsuya Gima, Yasuaki Kobayashi, and Yuto Okada 16:1–16:17

Visualizing Treewidth
Alvin Chiu, Thomas Depian, David Eppstein, Michael T. Goodrich, and Martin Nöllenburg 17:1–17:20

Internally-Convex Drawings of Outerplanar Graphs in Small Area
Michael A. Bekos, Giordano Da Lozzo, Fabrizio Frati, Giuseppe Liotta, and Antonios Symvonis 18:1–18:17

Crossing and Non-Crossing Families
Todor Antić, Martin Balko, and Birgit Vogtenhuber 19:1–19:15

A Systematic Approach to Crossing Numbers of Cartesian Products with Paths
Zayed Asiri, Ryan Burdett, Markus Chimani, Michael Haythorpe, Alex Newcombe, and Mirko H. Wagner 20:1–20:32

On Geometric Bipartite Graphs with Asymptotically Smallest Zarankiewicz Numbers
Parinya Chalermsook, Ly Orgo, and Minoo Zarsav 21:1–21:24

NNP-NET: Accelerating t-SNE Graph Drawing for Very Large Graphs by Neural Networks
Ilan Hartskeerl, Tamara Mchedlidze, Simon van Wageningen, Peter Vangorp, and Alexandru Telea 22:1–22:22

The Price of Connectivity Augmentation on Planar Graphs
Hugo A. Akitaya, Justin Dallant, Erik D. Demaine, Michael Kaufmann, Linda Kleist, Frederick Stock, Csaba D. Tóth, and Torsten Ueckerdt 23:1–23:24

String Graph Obstacles of High Girth and of Bounded Degree
Maria Chudnovsky, David Eppstein, and David Fischer 24:1–24:18

Characterizing and Recognizing Twistedness
Oswin Aichholzer, Alfredo García, Javier Tejel, Birgit Vogtenhuber, and Alexandra Weinberger 25:1–25:17

1-Planar Unit Distance Graphs with More Edges Than Matchstick Graphs
Eliška Červenková and Jan Kratochvíl 26:1–26:10

Geometry Matters in Planar Storyplans
Alexander Dobler, Maximilian Holzmüller, and Martin Nöllenburg 27:1–27:9

Treewidth of Outer k-Planar Graphs
Rafał Pyzik 28:1–28:16

Stabbing Faces by a Convex Curve
David Eppstein 29:1–29:8

Universal Quality Metrics for Graph Drawings: Which Graphs Excite Us Most?
Gavin J. Mooney, Tim Hegemann, Alexander Wolff, Michael Wybrow, and Helen C. Purchase 30:1–30:20

From Local Pair-Crossing Number to Local Crossing Number
Jacob Fox, János Pach, and Andrew Suk 31:1–31:6

Planar Stories of Graph Drawings: Algorithms and Experiments
Carla Binucci, Sabine Cornelsen, Walter Didimo, Seok-Hee Hong, Eleni Katsanou, Maurizio Patrignani, Antonios Symvonis, and Samuel Wolf 32:1–32:19

Edge Densities of Drawings of Graphs with One Forbidden Cell
Benedikt Hahn, Torsten Ueckerdt, and Birgit Vogtenhuber 33:1–33:17

Layered Polyline Drawings of Planar Graphs
Debajyoti Mondal 34:1–34:10

A Walk on the Wild Side: A Shape-First Methodology for Orthogonal Drawings
Giordano Andreola, Susanna Caroppo, Giuseppe Di Battista, Fabrizio Grosso, Maurizio Patrignani, and Allegra Strippoli 35:1–35:20

Reconfiguration in Curve Arrangements to Reduce Self-Intersections and Popular Faces
Florestan Brunck, Hsien-Chih Chang, Maarten Löffler, Tim Ophelders, and Lena Schlipf 36:1–36:18

Show Me Your Best Side: Characteristics of User-Preferred Perspectives for 3D Graph Drawings
Lucas Joos, Gavin J. Mooney, Maximilian T. Fischer, Daniel A. Keim, Falk Schreiber, Helen C. Purchase, and Karsten Klein 37:1–37:19

Stress in Graph Drawings: Perception, Preference, and Performance
Gavin J. Mooney, Jacob Miller, Michael Wybrow, Stephen Kobourov, and Helen C. Purchase 38:1–38:23

Optimizing Wiggle in Storylines
Alexander Dobler, Tim Hegemann, Martin Nöllenburg, and Alexander Wolff 39:1–39:17

An Algorithm for Accurate and Simple-Looking Metaphorical Maps
Eleni Katsanou, Tamara Mchedlidze, Antonios Symvonis, and Thanos Tolias 40:1–40:17

Graph Drawing Contest Report

Graph Drawing Contest Report
Sara Di Bartolomeo, Fabian Klute, Debajyoti Mondal, and Jules Wulms 41:1–41:11

Graph Drawing Contest Abstracts

Journey of a Time Machine
Florian Sass, Jakob Speitkamp, and Guilherme Monteiro Oliveira 42:1–42:2

Winning the GD Challenge for the 4th Time: Our Approach
Julien Bianchetti and Laurent Moalic 43:1–43:4

Poster Abstracts

Recovering Graphs from Their Witness Unit Square Representation
Maarten Löffler, Frank Staals, and Soeren Terziadis 44:1–44:4

TReView: Visualizing the European Union Transparency Register
Cristiano Bernardini, Davide Campanelli, Walter Didimo, Luca Grilli, Giuseppe Liotta, and Benedetto Ponti 45:1–45:5

Counting Triangulations of Fixed Cardinal Degrees
Erin Chambers, Tim Ophelders, Anna Schenfisch, and Julia Sollberger 46:1–46:4

Reconfigurations of Plane Caterpillars and Paths
Todor Antić, Guillermo Gamboa Quintero, and Jelena Glišić 47:1–47:5

Drawing Trees and Cacti with Integer Edge Lengths on a Polynomial-Size Grid
Henry Förster, Stephen Kobourov, Jacob Miller, and Johannes Zink 48:1–48:4

Defective Linear Layouts of Graphs
Michael A. Bekos, Carla Binucci, Emilio Di Giacomo, Walter Didimo, Luca Grilli, Maria Eleni Pavlidi, Alessandra Tappini, and Alexandra Weinberger . 49:1–49:4

Reeb Lobsters Are 1-Planar
Maarten Löffler, Miriam Münch, and Ignaz Rutter 50:1–50:5

Graph Tiles
Oswin Aichholzer, Robert Ganian, Phillip Keldenich, Maarten Löffler, Gert Meijer, Alexandra Weinberger, and Carola Wenk 51:1–51:5

Investigating Crossing Perception in 3D Graph Visualisation
Ying Zhang, Niklas Gröne, Giuseppe Liotta, Falk Schreiber, and Karsten Klein ... 52:1–52:5

Edge Bundling as a Multi-Objective Optimization Problem
Raissa dos Santos Vieira, Hugo A. D. do Nascimento, Joelma de Moura Ferreira, Les Foulds, Karsten Klein, and Falk Schreiber 53:1–53:5

BH-tsNET, FIt-tsNET, L-tsNET: Fast tsNET Algorithms for Large Graph Drawing
Amyra Meidiana, Seok-Hee Hong, and Kwan-Liu Ma 54:1–54:5

EnMRgy: Energy Network Analysis in Mixed Reality
Lucas Joos, Maximilian T. Fischer, Alexander Frings, and Daniel A. Keim 55:1–55:5

Using Reinforcement Learning to Optimize the Global and Local Crossing Number
Timo Brand, Henry Förster, Stephen Kobourov, Robin Schukrafft, Markus Wallinger, and Johannes Zink 56:1–56:4

Preface

This volume contains the papers presented at GD 2025, the 33rd International Symposium on Graph Drawing and Network Visualization, held during September 24–26, 2025 in Norrköping, Sweden. Graph Drawing is concerned with the geometric representation of graphs and constitutes the algorithmic core of Network Visualization. Graph Drawing and Network Visualization are motivated by applications where it is crucial to visually analyze and interact with relational datasets. Examples of such application areas include data science, social sciences, Web computing, information systems, biology, geography, business intelligence, information security and software engineering.

The symposium on Graph Drawing and Network Visualization has been the main annual event in this area for more than 30 years. Its focus is on combinatorial and algorithmic aspects of graph drawing as well as the design of network visualization systems and interfaces. Information about the conference series and past symposia is maintained at `http://www.graphdrawing.org`.

A total of 94 participants from 18 different countries attended the conference. Regular papers could be submitted to one of two distinct tracks: Track 1 for papers on fundamental theoretical graph drawing advances, such as on combinatorial and algorithmic aspects, and Track 2 for papers on practical aspects of graph drawing, such as experimental, applied, and network visualization aspects. Short papers were given a separate category, which welcomed both theoretical and applied contributions. An additional track was devoted to poster submissions.

All the tracks were handled by a single Program Committee. As in previous editions of GD, the papers in the different tracks did not compete directly with each other, but all Program Committee members were invited to review papers from any track in a "lightweight double-blind" process.

In response to the call for papers, the Program Committee received a total of 111 submissions, consisting of 96 papers (38 in Track 1, 33 in Track 2, and 25 in the short paper category) and 15 posters. More than 300 reviews were provided, about 40% having been contributed by external sub-reviewers. After extensive electronic discussions by the Program Committee via EasyChair, interspersed with virtual meetings of the Program Chairs producing incremental accept/reject proposals, 31 long papers, 7 short papers, and 13 posters were selected for inclusion in the scientific program of GD 2025. This resulted in an overall paper acceptance rate (not considering posters) of 39.6% (50.0% in Track 1, 36.4% in Track 2, and 28% in the short paper category). As is common in GD, some hard choices had to be made in particular during the final acceptance/rejection round, where several papers that clearly had merit still did not make the cut. However, the large number of high-quality submissions reflects the strength of the community.

For the second time in GD history, the proceedings are published by LIPIcs. The excellent support by the Dagstuhl LIPIcs team and by the GD 2024 Program Chairs helped to streamline the process.

There were two invited talks at GD 2025. Hans Bodlaender from Utrecht University (Netherlands) gave a survey about several central notions from parameterized complexity and discussed some recent developments, including the classes XNLP and XALP. Huamin Qu from Hong Kong University of Science and Technology (HKUST) discussed how to transform graph visualization through AI and human-AI Collaboration. Abstracts of both invited talks are included in these proceedings.

33rd International Symposium on Graph Drawing and Network Visualization (GD 2025).
Editors: Vida Dujmović and Fabrizio Montecchiani
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The conference gave out best paper awards in Track 1 and Track 2, as well as a best presentation award and a best poster award. The award for the best paper in Track 1 was given to "Characterizing and Recognizing Twistedness" by Oswin Aichholzer, Alfredo Garcia, Javier Tejel, Birgit Vogtenhuber and Alexandra Weinberger, and the award for the best paper in Track 2 was given to "OOPS: Optimized One-Planarity Solver via SAT" by Sergey Pupyrev. Based on a majority vote of the conference participants, the best presentation award was given to Benedikt Hahn for the presentation of "Edge densities of drawings of graphs with one forbidden cell" and the best poster award was given to Cristiano Bernardini, Davide Campanelli, Walter Didimo, Luca Grilli, Giuseppe Liotta and Benedetto Ponti for the poster titled "TReView: Visualizing the European Union Transparency Register".

A PhD School was held on the two days prior to the conference. Four half-day sessions led by Sara Di Bartolomeo, Markus Chimani, Daniel Archambault, and Camilla Forsell covered both theoretical and practical topics in graph drawing and network visualization.

As is traditional, the 32nd Annual Graph Drawing Contest was held during the conference. The contest was divided into two parts, creative topics and the live challenge. The creative topics task featured a single graph representing the events and characters from the Netflix series Dark, which centers on time travel. Since a special dataset requires a special setting, the submissions received for the contest were displayed in the Norrköping Decision Arena, a state-of-the-art facility featuring a 360-degree screen allowing the audience to immerse themselves in the visualizations. The live challenge focused on minimizing the local crossing number of drawings on a restricted integer grid. There were two categories: manual and automatic. We thank the Contest Committee, chaired by Fabian Klute, for preparing interesting and challenging problems. A report on the contest is included in these proceedings.

Many people and organizations contributed to the success of GD 2025. We would like to thank all members of the Program Committee and the external reviewers for carefully reviewing and discussing the submitted papers and posters; this was crucial for putting together a strong and interesting program. We would also like to thank all authors who chose GD 2025 as the publication venue for their research.

We are grateful for the support of our "Platinum" sponsor Carl Tryggers Foundation for Scientific Research, our "Gold" sponsors Tom Sawyer Software and yWorks, our "Silver" sponsor Visual Sweden, and our contributors Visualiseringscenter C and Linköping University. Their generous support helped to ensure the continued success of this conference. Furthermore, we would like to thank the LIPIcs team for their support during the creation of these proceedings.

Our special thanks go to all the members of the organizing committee based at Linköping University, in particular to the co-chairs Andreas Kerren and Kostiantyn Kucher.

The 34th International Symposium on Graph Drawing and Network Visualization (GD 2026) will take place during August 19-21, 2026, in St. Catharines, ON, Canada and will be hosted by Brock University. Debajyoti Mondal and Rahnuma Islam Nishat will co-chair the Organizing Committee, and Maarten Löffler and Silvia Miksch will co-chair the Program Committee.

October 2025 Vida Dujmović and Fabrizio Montecchiani

Sponsors

Platinum Sponsor



CARL TRYGGERS
STIFTELSE
FÖR VETENSKAPLIG FORSKNING

Gold Sponsors



yworks
the diagramming experts



Tom Sawyer
SOFTWARE

Silver Sponsor



VISUAL
SWEDEN

33rd International Symposium on Graph Drawing and Network Visualization (GD 2025).
Editors: Vida Dujmović and Fabrizio Montecchiani
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

List of Authors

Hugo A. Akitaya (23)
University of Massachusetts Lowell, MA, USA

Oswin Aichholzer (5, 12, 25, 51)
Graz University of Technology, Austria

Jawaherul Md. Alam (9)
Amazon Inc., Tempe, AZ, USA

Giordano Andreola (35)
Roma Tre University, Italy

Todor Antić (10, 19, 47)
Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Zayed Asiri (20)
Flinders University, Adelaide, Australia

Martin Balko (19)
Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Michael A. Bekos (6, 9, 18, 49)
Department of Mathematics, University of Ioannina, Greece

Cristiano Bernardini (45)
University of Perugia, Italy

Julien Bianchetti (43)
Université de Haute-Alsace, IRIMAS UR 7499, F-68100 Mulhouse, France

Carla Binucci (32, 49)
University of Perugia, Italy

Hans L. Bodlaender (1)
Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands

Timo Brand (56)
Technical University of Munich, Germany

Sofia Brenner (12)
University of Kassel, Germany

Florestan Brunck (36)
University of Copenhagen, Denmark

Andreas Bulling (11)
University of Stuttgart, Germany

Ryan Burdett (20)
Flinders University, Adelaide, Australia

Davide Campanelli (45)
University of Perugia, Italy

Susanna Caroppo (35)
Roma Tre University, Italy

Parinya Chalermsook (21)
University of Sheffield, England

Erin Chambers (46)
Department of Computer Science and Engineering, University of Notre Dame, IN, USA

Hsien-Chih Chang (36)
Dartmouth College, Hanover, NH, USA

Markus Chimani (20)
Theoretical Computer Science, Osnabrück University, Germany

Alvin Chiu (17)
University of California, Irvine, CA, USA

Maria Chudnovsky (24)
Mathematics Department, Princeton University, NJ, USA

Sabine Cornelsen (32)
University of Konstanz, Germany

Giordano Da Lozzo (18)
Roma Tre University, Italy

Justin Dallant (23)
Aarhus University, Denmark

Erik D. Demaine (23)
CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA

Thomas Depian (17)
TU Wien, Austria

Sara Di Bartolomeo (41)
TU Wien, Austria

Giuseppe Di Battista (8, 35)
Roma Tre University, Italy

Emilio Di Giacomo (49)
University of Perugia, Italy

Walter Didimo (32, 45, 49)
University of Perugia, Italy

Alexander Dobler (27, 39)
TU Wien, Austria

33rd International Symposium on Graph Drawing and Network Visualization (GD 2025).
Editors: Vida Dujmović and Fabrizio Montecchiani
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Joseph Dorfer (5, 12)
Graz University of Technology, Austria

David Eppstein (17, 24, 29)
University of California, Irvine, CA, USA

Joelma de Moura Ferreira (53)
IPOG Instituto de Pós-Graduação e Graduação, Goiânia, Brazil

Simon D. Fink (4)
Algorithms and Complexity Group, TU Wien, Austria

David Fischer (24)
Mathematics Department, Princeton University, NJ, USA

Maximilian T. Fischer (37, 55)
University of Konstanz, Germany

Les Foulds (53)
Universidade Federal de Goiás, Goiânia, Brazil

Jacob Fox (31)
Department of Mathematics, Stanford University, CA, USA

Fabrizio Frati (18)
Roma Tre University, Italy

Alexander Frings (55)
University of Konstanz, Germany

Henry Förster (48, 56)
Technical University of Munich, Germany

Guillermo Gamboa Quintero (47)
Computer Science Institute of Charles University, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Robert Ganian (51)
TU Wien, Austria

Alfredo García (25)
Departamento de Métodos Estadísticos and IUMA, University of Zaragoza, Spain

Tatsuya Gima (16)
Hokkaido University, Sapporo, Japan

Jelena Glišić (47)
Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Miriam Goetze (15)
Karlsruhe Institute of Technology, Germany

Michael T. Goodrich (17)
University of California, Irvine, CA, USA

Luca Grilli (45, 49)
University of Perugia, Italy

Martin Gronemann (9)
Independent Researcher, Dortmund, Germany

Fabrizio Grosso (35)
CeDiPa - University of Perugia, Italy

Niklas Gröne (52)
University of Konstanz, Germany

Carolina Haase (13)
Trier University, Germany

Benedikt Hahn (33)
Graz University of Technology, Austria

Ilan Hartskeerl (22)
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Michael Haythorpe (20)
Flinders University, Adelaide, Australia

Tim Hegemann (30, 39)
Universität Würzburg, Germany

Hung P. Hoang (12)
TU Wien, Austria

Michael Hoffmann (15)
Department of Computer Science, ETH Zürich, Switzerland

Maximilian Holzmüller (27)
TU Wien, Austria

Seok-Hee Hong (32, 54)
University of Sydney, Australia

Vit Jelínek (10)
Computer Science Institute, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Lucas Joos (37, 55)
University of Konstanz, Germany

Eleni Katsanou (32, 40)
National Technical University of Athens, Greece

Michael Kaufmann (6, 9, 23)
Department of Computer Science, University of Tübingen, Germany

Daniel A. Keim (37, 55)
University of Konstanz, Germany

Phillip Keldenich (51)
TU Braunschweig, Germany

Wilhelm Kerle-Malcharek (11)
University of Konstanz, Germany

Philipp Kindermann (13)
Trier University, Germany

Karsten Klein (11, 37, 52, 53)
University of Konstanz, Germany

Linda Kleist (23)
Department of Informatics, Universität Hamburg, Germany

Fabian Klute (41)
Universitat Politècnica de Catalunya, Spain

Yasuaki Kobayashi (16)
Hokkaido University, Sapporo, Japan

Stephen Kobourov (38, 48, 56)
Technical University of Munich, Heilbronn, Germany

Jan Kratochvíl (26)
Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Tomasz Krawczyk (3)
Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland

William Lenhart (13)
Williams College, Williamstown, MA, USA

Giuseppe Liotta (8, 13, 18, 45, 52)
University of Perugia, Italy

Maarten Löffler (36, 44, 50, 51)
Department of Information and Computing Sciences, Utrecht University, The Netherlands; Department of Computer Science, Tulane University, New Orleans, LA, USA

Kwan-Liu Ma (54)
University of California at Davis, CA, USA

Tamara Mchedlidze (7, 22, 40)
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Amyra Meidiana (54)
The University of Sydney, Australia

Gert Meijer (51)
Stenden University of Applied Sciences, Leeuwarden, The Netherlands

Jacob Miller (38, 48)
Technical University of Munich, Heilbronn, Germany

Laurent Moalic (43)
Université de Haute-Alsace, IRIMAS UR 7499, F-68100 Mulhouse, France

Debajyoti Mondal (34, 41)
Department of Computer Science, University of Saskatchewan, Saskatoon, Canada

Guilherme Monteiro Oliveira (42)
TU Wien, Austria

Gavin J. Mooney (30, 37, 38)
Monash University, Melbourne, Australia

Miriam Münch (4, 50)
Faculty of Computer Science and Mathematics, University of Passau, Germany

Hugo A. D. do Nascimento (53)
Universidade Federal de Goiás, Goiânia, Brazil

Alex Newcombe (20)
Flinders University, Adelaide, Australia

Martin Nöllenburg (17, 27, 39)
TU Wien, Austria

Yuto Okada (16)
Nagoya University, Japan

Tim Ophelders (36, 46)
Department of Information and Computing Sciences, Utrecht University, The Netherlands; Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Ly Orgo (21)
Aalto University, Finland

János Pach (31)
HUN-REN Alfréd Rényi Institute of Mathematics, Budapest, Hungary

Maurizio Patrignani (8, 32, 35)
Roma Tre University, Italy

Maria Eleni Pavlidi (49)
University of Ioannina, Greece

Martin Pergel (10)
Department of Software and Computer Science Education, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Daniel Perz (12)
University of Perugia, Italy

Maximilian Pfister (6)
Department of Computer Science, University of Tübingen, Germany

Matthias Pfretzschner (4)
Faculty of Computer Science and Mathematics, University of Passau, Germany

Benedetto Ponti (45)
University of Perugia, Italy

Sergey Pupyrev (14)
Menlo Park, CA, USA

Helen C. Purchase (30, 37, 38)
Monash University, Melbourne, Australia

Rafał Pyzik (28)
Institute of Theoretical Computer Science, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland

Huamin Qu (2)
Hong Kong University of Science and Technology (HKUST), Hong Kong

Christian Rieck (12)
University of Kassel, Germany

Ignaz Rutter (4, 15, 50)
Faculty of Computer Science and Mathematics, University of Passau, Germany

Florian Sass (42)
TU Wien, Austria

Anna Schenfisch (46)
Department of Mathematics, KTH Royal Institute of Technology, Stockholm, Sweden

Lena Schlipf (36)
AI Center, Department of Computer Science, University of Tübingen, Germany

Falk Schreiber (11, 37, 52, 53)
University of Konstanz, Germany; Monash University, Melbourne, Australia

Felix Schröder (10)
Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Robin Schukrafft (56)
Technical University of Munich, Germany

Julia Sollberger (46)
Department of Mathematics, Vrije Universiteit Amsterdam, The Netherlands

Jakob Speitkamp (42)
TU Wien, Austria

Frank Staals (44)
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Frederick Stock (23)
University of Massachusetts Lowell, MA, USA

Allegra Strippoli (35)
Roma Tre University, Italy

Peter Stumpf (10)
Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic; Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague, Czech Republic

Andrew Suk (31)
Department of Mathematics, University of California San Diego, La Jolla, CA, USA

Antonios Symvonis (8, 18, 32, 40)
School of Applied Mathematical & Physical Sciences, National Technical University of Athens, Greece

Alessandra Tappini (49)
University of Perugia, Italy

Javier Tejel (25)
Departamento de Métodos Estadísticos and IUMA, University of Zaragoza, Spain

Alexandru Telea (22)
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Alexandru C. Telea (7)
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Soeren Terziadis (44)
Algorithms Group, TU Eindhoven, The Netherlands

Thanos Tolias (40)
National Technical University of Athens, Greece

Ioannis G. Tollis (8)
Department of Computer Science, University of Crete, Heraklion, Greece

Csaba D. Tóth (23)
Department of Mathematics, California State University Northridge, Los Angeles, CA, USA; Department of Computer Science, Tufts University, Medford, MA, USA

Torsten Ueckerdt (15, 23, 33)
Karlsruhe Institute of Technology, Germany

Pavel Valtr (10)
Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Simon van Wageningen (7, 22)
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Peter Vangorp (22)
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Francesco Verciani (12)
University of Kassel, Germany

Raissa dos Santos Vieira (53)
Universidade Federal de Goiás, Goiânia, Brazil

Birgit Vogtenhuber (5, 19, 25, 33)
Graz University of Technology, Austria

Mirko H. Wagner (20)
Theoretical Computer Science, Osnabrück University, Germany

Markus Wallinger (56)
Technical University of Munich, Germany

Yao Wang (11)
University of Stuttgart, Germany

Alexandra Weinberger (25, 49, 51)
FernUniversität in Hagen, Germany

Carola Wenk (51)
Tulane University, New Orleans, LA, USA

Samuel Wolf (32)
University of Würzburg, Germany

Alexander Wolff (30, 39)
Universität Würzburg, Germany

Jules Wulms (41)
TU Eindhoven, The Netherlands

Michael Wybrow (30, 38)
Monash University, Melbourne, Australia

Minoo Zarsav (21)
Aalto University, Finland

Lin Zhang (11)
University of Stuttgart, Germany

Ying Zhang (11, 52)
University of Konstanz, Germany

Johannes Zink (48, 56)
Technical University of Munich, Germany

Eliška Červenková (26)
Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

A Sketch of Parameterized Complexity

Hans L. Bodlaender
Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands

Abstract

In the field of parameterized complexity, we study algorithms for and the complexity of problems where one part of the input is a parameter that is assumed to be small. In this talk, a survey will be given of several central notions from parameterized complexity, and discuss some recent developments, including the classes XNLP and XALP. These topics will be illustrated with examples from results on graph layout and graph drawing.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Fixed parameter tractability

Keywords and phrases Parameterized complexity, XNLP, XALP

Digital Object Identifier 10.4230/LIPIcs.GD.2025.1

Category Invited Talk



© Hans L. Bodlaender;
licensed under Creative Commons License CC-BY 4.0
33rd International Symposium on Graph Drawing and Network Visualization (GD 2025).
Editors: Vida Dujmović and Fabrizio Montecchiani; Article No. 1; pp. 1:1–1:1

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

Transforming Graph Visualization through AI and Human-AI Collaboration

Huamin Qu
Hong Kong University of Science and Technology (HKUST), Hong Kong

Abstract

In recent years, the intersection of artificial intelligence (AI) and graph visualization has led to advancements that enhance our ability to analyze and interpret complex data. In this talk, I will explore how AI and human-AI collaboration have transformed graph visualization, focusing on three key themes: efficiency in graph visualization, the integration of data storytelling, and the creative potential of human-AI partnerships.

In the first part of my talk, I will discuss how AI has been employed to create more efficient graph visualizations. I will highlight our innovative deep learning-based method for assessing the readability of graph layouts directly from images. This approach overcomes the limitations of traditional readability metrics, allowing for a more efficient evaluation of graph aesthetics, particularly in dense networks.

Next, I will delve into the application of graph visualization in data storytelling and virtual reality (VR) environments. I will present how tangible interactions can enhance live presentations of network visualizations, showcasing the effectiveness of intuitive physical interactions in engaging audiences. Additionally, I will discuss the development of semi-automatic data tours that guide users through complex networks, making exploration more intuitive and less time-consuming.

In the final section of my talk, I will focus on the creative aspects of human-AI collaboration in graph visualization. I will examine how generative AI techniques are reshaping the roles of humans and AI in the storytelling process, discussing the shift from human creators to AI-assisted storytelling. This evolution leads to innovative visualization techniques and highlights emerging collaboration patterns that enhance the storytelling experience.

By addressing these themes, my talk will illustrate the impact of AI and human-AI collaboration on graph visualization, highlighting both the opportunities and challenges that lie ahead in this rapidly evolving field.

2012 ACM Subject Classification Human-centered computing → Visualization

Keywords and phrases Graph Visualization, Virtual Reality, Human-AI Collaboration

Digital Object Identifier 10.4230/LIPIcs.GD.2025.2

Category Invited Talk

© Huamin Qu;
licensed under Creative Commons License CC-BY 4.0
33rd International Symposium on Graph Drawing and Network Visualization (GD 2025).
Editors: Vida Dujmović and Fabrizio Montecchiani; Article No. 2; pp. 2:1–2:1
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

On the Structure of Normalized Models of Circular-Arc Graphs I. Hsu's Approach

Tomasz Krawczyk
Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland

Abstract

In the work [$\mathcal{O}(m \cdot n)$ *algorithms for the recognition and isomorphism problems on circular-arc graphs*, *SIAM J. Comput. 24(3), 411–439, (1995)*], Wen-Lian Hsu claims three results concerning the class of circular-arc graphs:

- the design of so-called *decomposition trees* that represent the structure of all normalized intersection models of circular-arc graphs,
- an $\mathcal{O}(nm)$-time recognition algorithm for circular-arc graphs,
- an $\mathcal{O}(nm)$-time isomorphism algorithm for circular-arc graphs.

In [*Discrete Math. Theor. Comput. Sci., 15(1), 157–182, 2013*] Curtis, Lin, McConnell, Nussbaum, Soulignac, Spinrad, and Szwarcfiter showed that Hsu's isomorphism algorithm is incorrect. In this note, we show that the other two results – namely, the construction of decomposition trees and the recognition algorithm – are also flawed.

We also present the main ideas that made it possible to construct a data structure that maintains normalized models of circular-arc graphs.

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

Keywords and phrases intersection graphs and models, circular-arc graphs and circular-arc intersection models

Digital Object Identifier 10.4230/LIPIcs.GD.2025.3

Funding Partially supported by grant no. 2024/53/B/ST6/02558 from National Science Centre, Poland.

1 Introduction

One of the most important and widely studied models of geometric graph representation is the intersection model, in which vertices are represented by geometric sets and edges correspond to pairs of intersecting sets. Formally, the *intersection model* or the *representation* of a graph G within a family of geometrical objects $\mathcal{R}$ is a mapping $\phi : V(G) \to \mathcal{R}$ that maps the vertices of G into objects in $\mathcal{R}$ such that for every two distinct vertices u and v in G we have $uv \in E(G)$ if and only if $\phi(u) \cap \phi(v) \neq \emptyset$. Graphs which admit such a representation are called *intersection graphs* of objects from $\mathcal{R}$ or *$\mathcal{R}$-intersection graphs*. By considering families containing some specific kinds of objects we obtain different intersection graph classes, such as:

- *interval graphs*, which are intersection graphs of intervals on a line,
- *permutation graphs*, which are intersection graphs of segments spanned between two disjoint parallel lines,
- *circular-arc graphs*, which are intersection graphs of arcs of a circle,
- *circle graphs*, which are intersection graphs of chords of a circle.

Clearly, every interval graph is a circular-arc graph. Since permutation graphs can be equivalently defined as the intersection graphs of chords spanned between two disjoint arcs of a circle, every permutation graph is a circle graph. A graph G is a *comparability graph* if its edges can be oriented $\prec$ such that $(V(G), \prec)$ is a transitive and irreflexive relation


© Tomasz Krawczyk;
licensed under Creative Commons License CC-BY 4.0

on $V(G)$. A *co-comparability graph* is the complement of a comparability graph. Notably, every interval graph (permutation graph, respectively) G is a co-comparability graph. Indeed, given an intersection model of an interval (permutation, respectively) graph G, the pairs of non-intersecting intervals (segments, respectively) can be ordered naturally from left to right. This ordering induces a transitive orientation of the edges in the complement $\overline{G}$ of G.

For a given intersection graph class, a fundamental combinatorial problem is to characterize all intersection models of a graph in that class and to design a data structure that efficiently represents all such models. Such data structures play a central role in algorithmic applications, where the task is to construct, for a given abstract graph specified by its vertices and edges, an intersection model satisfying prescribed criteria (see, e.g., *partial representation extension* problems). When these data structures also possess a tree-like structure, they can be leveraged to design graph isomorphism algorithms: testing whether two graphs are isomorphic then reduces to checking isomorphism between the tree-like data structures representing their intersection models. For each graph class introduced above – except for circular-arc graphs – such a data structure is known: we have *PQ-trees* for interval graphs, *modular decomposition trees* for permutation graphs, and *split decomposition trees* for circle graphs.

In the work [*$\mathcal{O}(m \cdot n)$ algorithms for the recognition and isomorphism problems on circular-arc graphs*, *SIAM J. Comput. 24(3), 411–439, (1995)*] [5], Wen-Lian Hsu claims three results concerning the class of circular-arc graphs. The first claims a construction of a *decomposition tree*, a data structure intended to represent the set of all *normalized intersection models* of a circular-arc graph (an analogue of the PQ-tree for an interval graph). *Normalized intersection models* (formally defined in Section 2) of a circular-arc graph reflect the neighborhood relation between its vertices and can be seen as its canonical representations; in particular, any intersection model can be made normalized by possibly extending some of its arcs. Based on these decomposition trees, Hsu claims $\mathcal{O}(nm)$-time algorithms for both the recognition and isomorphism problem for circular-arc graphs.

In this work, we show that the description of the normalized intersection models given by Hsu in [5] is not correct. Specifically, we show the following:

▶ **Counterexample 1.** *There are circular-arc graphs whose all normalized models do not follow the description given by Hsu.*

As a consequence, the decomposition trees proposed by Hsu, along with his polynomial-time algorithms for recognizing and testing isomorphism of circular-arc graphs, are also incorrect. We note that Curtis, Lin, McConnell, Nussbaum, Soulignac, Spinrad, and Szwarcfiter [1] had already identified a different flaw in Hsu's isomorphism algorithm in 2013. Hsu's approach – based on the modular decomposition by Gallai (see Section 3) – determines the main steps required to describe the structure of normalized models of circular-arc graphs. In this paper, we identify several errors in Hsu's work, showing that some key steps of his work were incorrectly carried out.

Although Hsu's work [5] contains flaws, the overall approach it takes is appropriate. In [6], we modify several crucial definitions within Hsu's framework, which allows us to develop a data structure that represents all normalized models of circular-arc graphs. We want to mention that, despite the errors in Hsu's original work, his ideas had a strong influence on the work done in [6]. We refer the reader to [6] for more details.

Our paper is organized as follows:

- in Section 2 we introduce notation and basic tools used throughout the paper,
- in Section 3 we describe the prior research that influenced Hsu's work on the problem of characterizing normalized models of circular-arc graphs,

- in Section 4 we sketch the approach used by Hsu, we show the main errors in his work, and we prove Counterexample 1,
- in Section 5 we describe the main ideas used in [6] that allow, based on the same approach, to devise a data structure representing all normalized models of a circular-arc graph.

2 Preliminaries

All graphs discussed in this paper are finite and simple. The vertex set and edge set of a graph G are denoted by, respectively, $V(G)$ and $E(G)$. The *neighborhood* of a vertex v, denoted by $N_G(v)$, comprises vertices adjacent to v, i.e., $N_G(v) = \{u \in V(G) : uv \in E(G)\}$, and the *closed neighborhood* of v is $N_G[v] = N_G(v) \cup \{v\}$. A vertex $u \in V(G)$ is *universal* in G if $N_G[u] = V(G)$. Vertices $u, v \in V(G)$ are *twins* in G if $N_G[u] = N_G[v]$. For a subset $U \subseteq V(G)$, we denote by $G[U]$ the graph induced by the set U.

Let $(V(G), \prec)$ be a transitive orientation of a comparability graph G. For subsets $M_1, M_2 \subseteq V(G)$ we write $M_1 \prec M_2$ if we have $u \prec v$ for every $u \in M_1$ and $v \in M_2$.

2.1 Modular decomposition tree

The definitions that follow are due to Gallai [4].

Let G be a graph and let $\overline{G}$ be the complement of G. A non-empty set $M \subseteq V(G)$ is a *module* of G if every vertex outside M is adjacent to all or to none of the vertices in M. The singleton sets and the whole set $V(G)$ are the *trivial* modules of G. A module M of G is *strong* if $M \subseteq N$, $N \subseteq M$, or $M \cap N = \emptyset$ for every other module N in G. In particular, two strong modules of G are either nested or disjoint. The *modular decomposition tree* of G, denoted by $\mathcal{M}(G)$, consists of all strong modules of G. The set $\mathcal{M}(G)$, ordered by inclusion, forms a tree in which $V(G)$ is the root, the maximal proper strong submodules of a strong module M are the children of M (they form a partition of M), and the singleton modules $\{v\}$ for $v \in V(G)$ are the leaves. A strong module M in G is *parallel* if $G[M]$ is disconnected, *serial* if $\overline{G[M]}$ is disconnected, and *prime* if both $G[M]$ and $\overline{G[M]}$ are connected. If $G[M]$ (respectively, $\overline{G[M]}$) is disconnected, then the children of M in the modular decomposition tree of G are the connected components of $G[M]$ ($\overline{G[M]}$, respectively).

Now assume that G is a comparability graph. The relation between the transitive orientations of the graph G and the modular decomposition tree of G was described by Gallai [4].

▶ **Theorem 2** ([4]). *If modules M_1, M_2 in $\mathcal{M}(G)$ are such that $uv \in E(G)$ for any $u \in M_1$ and $v \in M_2$, then every transitive orientation $(V(G), \prec)$ of G satisfies either $M_1 \prec M_2$ or $M_2 \prec M_1$.*

For a strong module M in G let G_M denote the graph on M whose edge set $E(G_M)$ contains all the edges from G that join the vertices from two different children of M. If xy is an edge in G, then $xy \in E(G_M)$ for exactly one strong module $M \in \mathcal{M}(G)$. Hence, the set $\{E(G_M) : M \in \mathcal{M}(G)\}$ forms a partition of the edge set $E(G)$ of the graph G.

▶ **Theorem 3** ([4]). *There is a one-to-one correspondence between the set of transitive orientations $(V(G), \prec)$ of G and the families*

$$\{(M, \prec_M) : M \in \mathcal{M}(G) \text{ and } \prec_M \text{ is a transitive orientation of } G_M\}$$

given by $x \prec y \iff x \prec_M y$, where M is the module in $\mathcal{M}(G)$ such that $xy \in E(G_M)$.

The above theorem asserts that every transitive orientation of G restricted to the edges of the graph G_M induces a transitive orientation of G_M, for every $M \in \mathcal{M}(G)$, and that every transitive orientation of G can be obtained by independent transitive orientations of the graphs G_M, for $M \in \mathcal{M}(G)$. Gallai [4] characterized all possible transitive orientations of the graph G_M, where M is a module of $\mathcal{M}(G)$.

▶ **Theorem 4** ([4]). *Let M be a prime module in $\mathcal{M}(G)$. Then, G_M has two transitive orientations, one being the reverse of the other.*

If M is parallel, then $E(G_M) = \emptyset$, and the graph G_M has exactly one (empty) transitive orientation. If M is serial, the transitive orientations of G_M correspond to the total orderings of its children, that is, every transitive orientation of G_M is of the form $M_{i_1} \prec_M \ldots \prec_M M_{i_k}$, where $i_1 \ldots i_k$ is a permutation of $[k]$ and $M_1, \ldots, M_k$ are the children of M in $\mathcal{M}(G)$.

A graph G is *prime* if every module of G is trivial. By Theorem 4, every prime comparability graph admits a unique (up to reversal) transitive orientation.

2.2 Join decomposition

Since our counterexamples involve graphs that admit "join decompositions", we reproduce the following definitions exactly as stated in Hsu's paper [5] (see page 417, lines –7 to –1).

A graph G is said to have a *join* if $V(G)$ can be partitioned into sets V_0, V_1, V_2, and V_3 with $|V_0 \cup V_1| \geqslant 2$ and $|V_2 \cup V_3| \geqslant 2$ so that every possible edge exists between V_1 and V_2, no edge exists between V_0 and $V_2 \cup V_3$, and no edge exists between V_3 and $V_0 \cup V_1$. In this case, we say that the graph G is *decomposable into graphs H_1 and H_2* where H_1 is the induced subgraph of $G[V_0 \cup V_1]$ with an extra vertex v adjacent to every vertex in V_1 and H_2 is the induced subgraph of $G[V_2 \cup V_3]$ with an extra vertex u adjacent to every vertex in V_2. If the above holds, then (V_0, V_1, V_2, V_3) is a *join* in G and H_1 and H_2 is a *decomposition* of G *induced by the join* (V_0, V_1, V_2, V_3). See Figures 6(c)-(d) for an illustration. A graph G is said to be *j-inseparable* if it does not contain a join.

2.3 Normalized models of circular-arc graphs

Let G be a circular-arc graph and let ψ be a circular-arc model of G such that all the arcs in the set $\{\psi(v) : v \in V(G)\}$ have distinct endpoints. We refer to Figure 1 which shows possible relations between two arcs in $\{\psi(v) : v \in V(G)\}$.

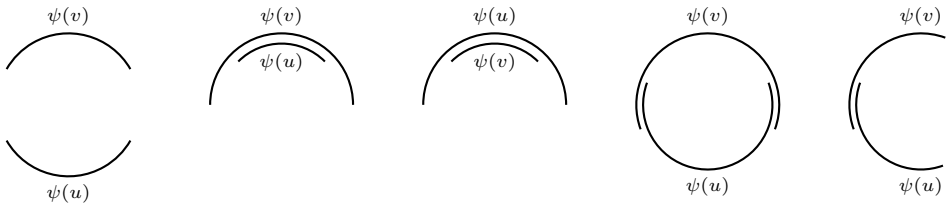


Figure 1 From left to right: $\psi(v)$ and $\psi(u)$ are disjoint, $\psi(v)$ contains $\psi(u)$, $\psi(v)$ is contained in $\psi(u)$, $\psi(v)$ and $\psi(u)$ cover the circle, and $\psi(v)$ and $\psi(u)$ overlap.

In so-called *normalized models*, defined in [5], the relative relation between the arcs reflects the closed neighbourhood relation between the vertices of G, as follows.

▶ **Definition 5.** *Let G be a circular-arc graph. A circular-arc model ψ of G is* normalized with respect to G *(shortly,* normalized*) if all the arcs from $\{\psi(v) : v \in V(G)\}$ have distinct endpoints and for every pair (v, u) of distinct vertices in G the following conditions are satisfied:*

1. *if* $uv \notin E(G)$*, then* $\psi(v)$ *and* $\psi(u)$ *are disjoint,*
2. *if* $N_G[u] \subsetneq N_G[v]$*, then* $\psi(v)$ *contains* $\psi(u)$*,*
3. *if* $N_G[v] \subsetneq N_G[u]$*, then* $\psi(v)$ *is contained in* $\psi(u)$*,*
4. *if* $N_G[v] \cup N_G[u] = V$*,* $N_G[w] \subsetneq N_G[v]$ *for every* $w \in N_G[v] \setminus N_G[u]$*, and* $N_G[w] \subsetneq N_G[u]$ *for every* $w \in N_G[u] \setminus N_G[v]$*, then* $\psi(v)$ *and* $\psi(u)$ *cover the circle,*
5. *If none of the above condition holds, then* $\psi(v)$ *and* $\psi(u)$ *overlap.*

Furthermore, for a pair (v, u) *of distinct vertices from* G*, we say that* v *and* u *are* disjoint *in* G*,* v contains u *in* G*,* v is contained *in* u *in* G*,* v *and* u cover the circle *in* G*, and* v *and* u overlap *in* G *if the pair* (v, u) *satisfies the assumption of statement* 1.*,* 2.*,* 3.*,* 4.*, and* 5.*, respectively.*

Note that the relation between vertices u and v is symmetric, with one exception: u is contained in v if and only if v contains u. It is known that if G has no universal vertices and no twins, every circular-arc model of G can be transformed into a normalized model by possibly extending some of its arcs [5].

3 Modular decomposition tree as a data structure representing intersection models of some geometric intersection graphs

One of the most powerful tools for designing data structures that maintain intersection models of geometric graph classes is the modular decomposition tree, introduced by Gallai [4] to characterize the structure of all transitive orientations of a comparability graph G – see Subsection 2.1 for details. In short, the structure of all transitive orientations of a comparability graph G satisfies the following properties:

(G1) If $U \subseteq V(G)$ is such that $G[U]$ is a prime subgraph of G, then $G[U]$ admits a unique transitive orientation (up to reversal).

(G2) The structure of all transitive orientations of G can be described (and represented) by means of the modular decomposition tree of G.

It turns out that the modular decomposition tree can be used to represent intersection models for various classes of geometric intersection graphs, in particular, permutation graphs and interval graphs.

Dushnik and Miller [2] proved that a graph G is a permutation graph if and only if G and its complement $\overline{G}$ admit transitive orientations. Furthermore, they showed that, if G is a permutation graph, then the intersection models of G correspond to the pairs of transitive orientations of G and $\overline{G}$. Formally, given an intersection model ϕ of G we can derive transitive orientations $\prec$ of G and $<$ of $\overline{G}$, as follows (see Figure 2):

$$\begin{array}{lcl} x \prec y & \iff & \phi(x) \text{ and } \phi(y) \text{ intersect and } \phi(x) \text{ precedes } \phi(y) \text{ in the upper line.} \\ x < y & \iff & \phi(x) \text{ is to the left of } \phi(y) \text{ in } \phi. \end{array} \tag{1}$$

On the other hand, given transitive orientations $\prec$ and $<$ of G and $\overline{G}$, respectively, one can construct a permutation model ϕ of G such that

$$\begin{array}{lcl} \phi(x) \text{ precedes } \phi(y) \text{ in the upper line} & \iff & x \prec y \text{ or } x < y, \\ \phi(x) \text{ precedes } \phi(y) \text{ in the lower line} & \iff & y \prec x \text{ or } x < y. \end{array} \tag{2}$$

As for interval graphs, a reader familiar with PQ-trees may observe that the PQ-tree of an interval graph G can be easily derived from the modular decomposition tree of G. In particular, Properties (G1) and (G2) have the following impact on the structure of the intersection models of a permutation/interval graph G:

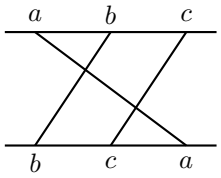


Figure 2 Intersection model ϕ of the permutation graph $G = (\{a, b, c\}, \{ab, ac\})$ corresponding to the transitive orientations $\{a \prec b, a \prec c\}$ and $\{b < c\}$ of G and $\overline{G}$, respectively.

(M1) If $U \subseteq V(G)$ is such that $G[U]$ is prime, then $G[U]$ admits a unique intersection model (up to certain normalizations and reflections).

(M2) The structure of the intersection models of G is represented by the modular decomposition tree of G.

Moreover, Theorem 2 asserts the following property of intersection models of permutation graphs:

(M3) In any segment intersection model of a permutation graph G, for every strong module M in G, the lower (respectively, upper) endpoints of the segments representing vertices in M lie within a contiguous interval on the lower (respectively, upper) line, containing no endpoints of segments corresponding to vertices outside M.

Regarding circular-arc graphs, Spinrad's work [7] was the first to describe the structure of normalized models for a specific subclass using the approach outlined above. Specifically, Spinrad focused on *co-bipartite* graphs, that is, graphs whose vertex set can be partitioned into two cliques.

Let us briefly outline the ideas of Spinrad. Let G be a *co-bipartite circular-arc* graph whose vertex set can be partitioned into two cliques, C_A and C_B. We assume that G has no universal vertices and no twins. Let G_c denote the graph on the vertex set $V(G)$ in which two vertices are joined by an edge if and only if they overlap according to Definition 5. In particular, in any normalized model of G, the arcs for u and v overlap exactly when uv is an edge of G_c. Let $\overline{G_c}$ be the complement of G_c. First, Spinrad proves that each model of G can be turned into a normalized model ψ such that all the arcs from $\{\psi(x) : x \in C_A\}$ contain the leftmost point A of the circle, and all the arcs from $\{\psi(x) : x \in C_B\}$ contain the rightmost point B of the circle. Let us call such models of G *strongly normalized* – see Figure 3 for an illustration. Note that no arc from $\{\psi(v) : v \in V(G)\}$ contains both the points A and B as G has no universal vertices. Hence, each arc from $\{\psi(v) : v \in V(G)\}$ has one endpoint in the upper semicircle and one endpoint in the lower semicircle. Next, Spinrad introduces an orientation $<$ on the edges of $\overline{G_c}$, as shown in Figure 3. Spinrad proves that $<$ is a transitive orientation of $\overline{G_c}$ and that $<$ can be represented by segments spanned between the upper and lower semicircle in such a way that the segment for u is to the left of the segment for v if and only if $u < v$ (note that that segments for u and v intersect if and only if vertices u and v overlap in G).

The above observations were used by Spinrad to devise a polynomial-time recognition algorithm for co-bipartite circular arc graphs [7]. Given a co-bipartite graph G as input, the algorithm constructs the overlap graph G_c of G and the orientation $<$ of the edges of $\overline{G_c}$. The graph G is accepted as a circular-arc graph if G_c is a permutation graph and $<$ is a transitive orientation of $\overline{G_c}$. Note that this approach allows us to conclude that the strongly normalized models of G correspond to the transitive orientations of the graph G_c. Indeed, G_c is a permutation graph and hence its models correspond to the pairs of transitive orientations

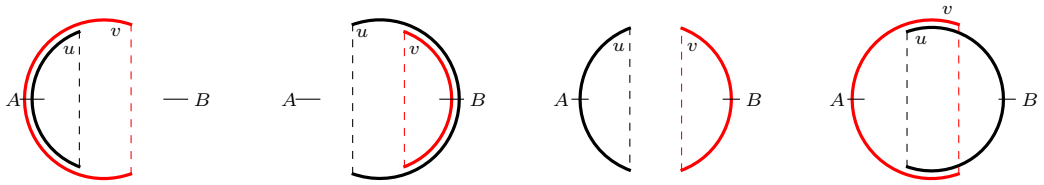


Figure 3 Spinrad sets $u < v$ if and only if either $u, v \in C_A$ and u is contained in v (left) or $u, v \in C_B$ and u contains v (middle left) or $u \in C_A$, $v \in C_B$, and u, v are disjoint (middle right), or $u \in C_B$, $v \in C_A$, and u, v cover the circle (right).

of G_c and $\overline{G_c}$. However, to obtain a model consistent with the relations between the vertices in G, non-intersecting segments need to be placed consistently with the $<$ relation. Summing up, Properties (M1)–(M3) assert that:

(S1) If $U \subseteq V(G)$ is such that $G_c[U]$ is prime, then $G[U]$ admits a unique strongly normalized model.

(S2) The set of all strongly normalized models of G is represented by the modular decomposition tree of the overlap graph G_c of G.

(S3) For every strong module M of G_c and every strongly normalized model of G, the lower (upper) endpoints of the arcs for the vertices in M lie within a contiguous arc on the lower (upper, respectively) semicircle, containing no endpoints of the arcs for the vertices from outside M.

4 Hsu's approach to the problem of characterizing normalized models of circular-arc graphs

To describe the structure of normalized models of circular-arc graphs, Hsu attempts to generalize Spinrad's ideas to the entire class of circular-arc graphs [5]. For this purpose, Hsu introduces the *overlap graph* G_c of G:

Definition [page 420, lines 11 to 12]. *Associate with each graph G the graph G_c that has the same vertex set as G such that two vertices in G_c are adjacent iff they are strictly but not strongly adjacent in G.*

In [5], two vertices u and v are said to be strictly but not strongly adjacent in G if and only if they overlap in G according to Definition 5. Then, Hsu transforms every normalized model R of G to a chord model D by converting each arc $R(v)$ into a chord $D(v)$ with the same endpoints as $R(v)$, for every $v \in V(G)$. Hsu observes that D is a chord model of the graph G_c, regardless of the choice of a normalized model R of G. Next, Hsu lists some properties of the chord models D of G_c derived from the normalized models R of G. To state them, for every vertex $v \in V(G)$ let I_v be the set of vertices not adjacent to v in G_c, and let

$$\begin{aligned} L_v &= \{u \in I_v : \quad u \text{ is contained in } v \text{ in } G \quad \text{or} \quad v \text{ and } u \text{ cover the circle in } G\}, \\ R_v &= \{u \in I_v : \quad u \text{ and } v \text{ are not adjacent in } G \quad \text{or} \quad u \text{ contains } v \text{ in } G\}. \end{aligned}$$

Hsu observes that for every $v \in V(G)$ the chords representing the vertices from the set L_v are on one side of $D(v)$ and the chords representing the vertices from the set R_v are on the other side of $D(v)$. See Figure 4 for an illustration. Based on this observation, Hsu assumes the following definition:

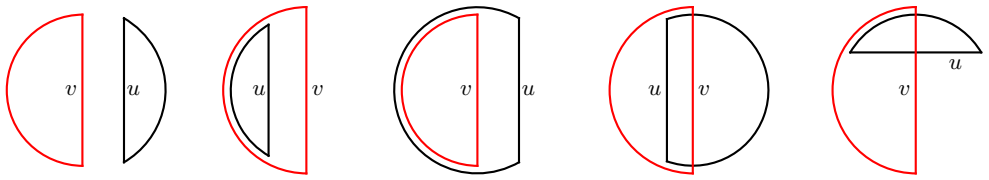


Figure 4 Relations between the arcs $R(v)$ and $R(u)$ and the corresponding chords $D(v)$ and $D(u)$ for the cases: v is disjoint with u, v contains u, v is contained in u, v and u cover the circle, and v and u overlap, respectively.

Definition [page 424, lines 18 to 20]. *A chord model D of G_c is* conformal *(to G) if for every vertex v of G the chords associated with vertices in L_v are on one side of the chord of $D(v)$ and those associated with vertices in R_v are on the other side of the chord of $D(v)$.*

Then, Hsu proves the correspondence between normalized models of G and the conformal models of G_c:

Theorem 5.6 [page 424, lines -18 to -17]. *Let G be a circular-arc graph [...]. Then a model for G_c is conformal if and only if it is a chord model corresponding to some normalized model for G.*

Eventually, Hsu tries to describe the structure of the chord models of G_c that are conformal to G. To this end, Hsu attempts to prove the properties similar to (S1) and (S2):

(H1) If $U \subseteq V(G)$ is such that $G_c[U]$ is prime, then G_c has a unique conformal model (up to reflection).

(H2) The structure of the conformal models of G_c can be described by the modular decomposition tree of G_c.

To accomplish Property (H2), Hsu introduces so-called *consistent modules* in the graph G_c (see Subsection 4.1.1 for more details) that were intended to satisfy the following properties:

(H3) For every conformal model D and every consistent module M of G_c there are two disjoint arcs A_M and B_M of the circle such that each arc A_M and B_M contains one endpoint of every chord for the vertices of M and no endpoint of any chord for the vertices from outside M (in particular, it means that $G_c[M]$ is a permutation graph). Furthermore, the maximal consistent modules in G_c form a partition of the set $V(G_c)$, and a possible placement of the arcs A_M and B_M for the maximal consistent modules M in the conformal models of G_c can be represented by the modular decomposition tree of G_c.

Note that when G is co-bipartite, Property (S3) asserts that every strong module of G_c is a consistent module of G_c. Moreover, the relative ordering of the endpoints of the maximal non-trivial consistent modules in the conformal models of G_c is represented by the modular decomposition tree of G_c.

First, we emphasize that Property (H1) is fundamental to the method, since the structure of conformal models is built upon the unique conformal models of the prime induced subgraphs of G_c. In this context, Property (H1) plays a role analogous to the uniqueness of transitive orientations of prime subgraphs in the characterization of all transitive orientations of a comparability graph. In Subsection 4.1 we show that Hsu's attempted proof of Property (H1) (Theorem 5.7 in [5]) is not correct – it relies on two claims, both of which are disproved in Section 4.1. However, the statement of Property (H1) holds true, which follows from the proof of an analogous property shown in [6] (proved for the oriented conformal models – see Section 5 for more details).

To establish Property (H2), Hsu divides the description of the structure of the conformal models of G_c into three cases depending on whether $V(G_c)$ is serial/prime/parallel in the modular decomposition tree of G_c. Hsu's work [5] correctly deals with the serial case (then, each component of $\overline{G_c}$ induces a co-bipartite circular-arc graph and we can use the results of Spinrad [7] to describe the normalized models of G), but is incorrect in the prime case, and is incomplete (and also incorrect) in the parallel case.

For the prime case, Hsu first attempts to prove Property (H1). Next, Hsu tries to partition the set $V(G_c)$ into "maximal consistent modules" and tries to prove that there is a unique circular order (up to reflection) in which the arcs A_M and B_M for maximal consistent modules M occur around the circle in the conformal models of G_c. In Subsection 4.1.1 we show that the "maximal consistent modules" determined by Hsu do not satisfy Property (H3). This enables us, in particular, to construct a circular-arc graph claimed by Counterexample 1. Hence, the theorems from [5] that establish Property (H3) for the "maximal consistent modules" determined by Hsu are also not correct – see Subsection 4.1.1 for more details.

Finally, the description of Hsu is also not correct in the parallel case. First, the errors mentioned in the prime case (partition into consistent modules) are transferred to the description of the conformal models of prime components of G_c (partition of the prime children of $V(G_c)$ into maximal consistent modules is not correct). Moreover, this part of Hsu's work seems to miss some important elements. For example, nowhere in his work we could find the description of maximal consistent modules for serial children of $V(G_c)$ – we refer to Section 4.2 for more details.

4.1 Prime graphs

When $V(G_c)$ forms a prime module in G_c, Hsu first considers the case where G_c is *s-inseparable*, which is equivalent to G_c being prime (see p. 416, lines 24–30). For this case, Hsu first attempts to prove the following:

Theorem 5.7 [page 425, lines 16 to 17]. *Let G be a circular-arc graph. If G_c is s-inseparable, then G has a unique normalized model.*

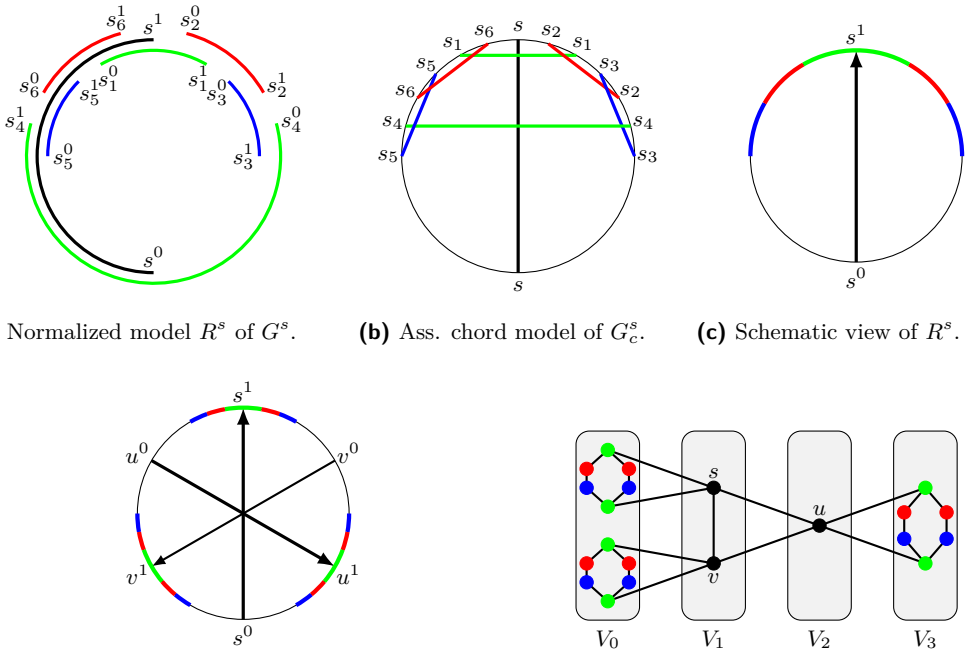
The Hsu's proof of Theorem 5.7 is split into two cases depending on whether G_c is j-inseparable (see Subsection 2.2). If G_c is j-inseparable, then G_c has a unique chord model, as proved by Gabor, Supowit, and Hsu [3], and hence G has a unique normalized model.

So, the interesting case is when G_c is not j-inseparable, that is, when G_c has a join (V_0, V_1, V_2, V_3). First, Hsu observes that $V_i \neq \emptyset$ for all $i = 0, \ldots, 3$ as G_c is s-inseparable. The remainder of the proof relies on two key claims, both of which we show to be false.

First, the proof of Theorem 5.7 tries to show the following property of the graph G_c and the join (V_0, V_1, V_2, V_3) of G_c:

Claim A [page 425, line -23 to -18]. *If $|V_1| > 0$ and there exists a vertex s in V_1 that is adjacent to all other vertices in $V_1 \cup V_2$, then it is easy to verify that $G_c \setminus \{s\}$ is s-inseparable. We can proceed with the construction below to show that $G_c \setminus \{s\}$ has a unique conformal model, and, furthermore, there is a unique way to insert the chord of s into this model. The argument is symmetric if such a vertex s belongs to V_2.*

We show that Claim A is false. To show a counterexample, we first define an auxiliary circular-arc graph G^s: the vertex set of G^s is $\{s, s_1, \ldots, s_6\}$, the edges of G^s can be read from the intersection model R^s of G^s shown in Figure 5(a). We leave the reader to verify that the model R^s is normalized (the vertices $s_1, \ldots, s_6$ induce a cycle in G^s and hence every two consecutive vertices on this cycle overlap in G^s). The associated circle graph G^s_c and the schematic view of the model R^s are illustrated in Figures 5(b)-(c).



(a) Normalized model R^s of G^s. **(b)** Ass. chord model of G_c^s. **(c)** Schematic view of R^s.

(d) Schematic view of the model R of G. **(e)** Graph G_c and its join decomposition.

Figure 5 Counterexample to Claim A.

Next, we define a circular-arc graph G by taking G^s and two isomorphic copies of G^s, say G^v and G^u, over the vertex sets $\{v, v_1, \ldots, v_6\}$ and $\{u, u_1, \ldots, u_6\}$, respectively. Again, we define the edges of G by providing an intersection model R of G. Given the schematic view of the model R, as shown in Figure 5(d), we construct the intersection model R for G as follows. We begin with the circular-arc model R^s of G^s and embed its arcs into the circle so that:

- The arc $R^s(s)$ shares its endpoints with the oriented chord s^0s^1 and lies to the left of this chord when traversed from s^0 to s^1.
- The endpoints of the remaining arcs in R^s are compressed so that they all fit within the colored arc adjacent to the head of the chord s^0s^1 (that is, the arc endpoints for $s_1, \ldots, s_6$, as well as the upper endpoint of s, are placed on the colored (blue–red–green) upper arc in the order shown in Figures 5(a) and 5(b)).

We paste in the same way the models R^v and R^u, which represent G^v and G^u, along the chords v^0v^1 and u^0u^1, respectively.

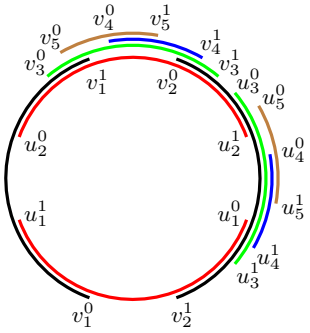
Once again, we leave it to the reader to verify that R is a normalized model of G. The associated circle graph G_c, along with its join decomposition (V_0, V_1, V_2, V_3), is shown in Figure 5(e). Note that:

- G_c is prime as G_c has no non-trivial modules,
- vertex s in V_1 is adjacent to all other vertices in $V_1 \cup V_2$.

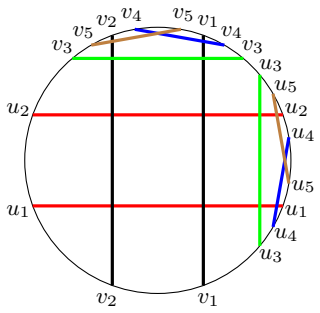
That is, G_c, (V_0, V_1, V_2, V_3), and s satisfy the assumptions of Claim A. The graph $G_c \setminus \{s\}$, however, is not prime as $G_c \setminus \{s\}$ is disconnected. So, Claim A is false.

Then, the proof assumes that G_c has a join (V_0, V_1, V_2, V_3) such that no vertex from $V_1 \cup V_2$ is adjacent to all other vertices in $V_1 \cup V_2$. Let H_1 and H_2 be the decomposition of G_c induced by (V_0, V_1, V_2, V_3). In this setting, the following property of H_1 and H_2 is claimed.

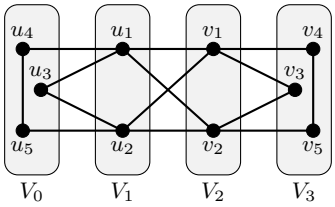
Claim B [page 425, line -19 to -16]. *Therefore, assume no such vertex s exists. Then it is easy to verify that the corresponding two component graphs, H_1 and H_2 of G_c (the existence of the substitution in either H_1 or H_2 would imply the existence of one in G_c), are s-inseparable.*



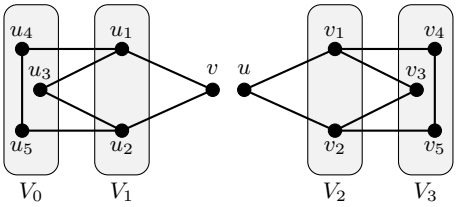
(a) Normalized circular-arc model R of G.



(b) Associated chord model D of G_c.



(c) Graph G_c and its join decomposition.



(d) Decomposition of G_c into H_1 and H_2.

Figure 6 Counterexample to Claim B.

Consider a circular-arc graph G whose normalized model R is shown in Figure 6(a). The associated circle graph G_c and its join (V_0, V_1, V_2, V_3) are shown in Figure 6(c), the chord model D associated with R is shown in Figure 6(b), and the decomposition H_1 and H_2 of G_c induced by the join (V_0, V_1, V_2, V_3) is shown in Figure 6(d). Note the following properties of G_c, H_1, and H_2:

- G_c is prime as G_c has no non-trivial modules and there is no vertex s in $V_1 \cup V_2$ adjacent to all other vertices in $V_1 \cup V_2$,
- H_1 contains a non-trivial module $\{u_3, v\}$ and H_2 contains a non-trivial module $\{u, v_3\}$.

In particular, Claim B is also false.

The proof of Theorem 5.4 by Hsu is concluded as follows. Since H_1 and H_2 are s-inseparable, they have unique chord models by induction. Eventually, there is a unique way to compose these models to get a conformal model of G_c.

4.1.1 Consistent modules

The precise definition of the consistent modules (they are also called as T-submodules of G_c is given in Section 6.1 of the Hsu's work [5]). Since it is very technical, we list only their properties as proven by Hsu.

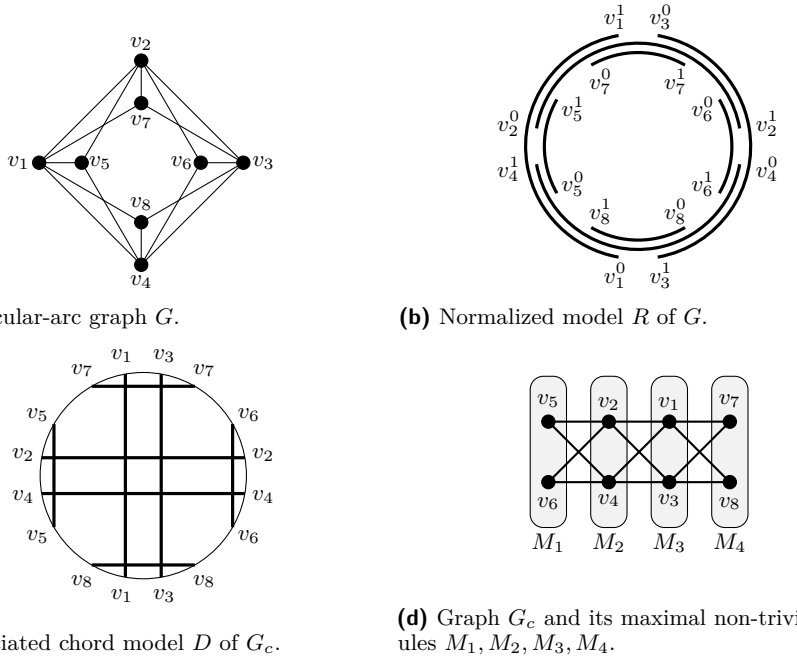
Lemma 6.5 [page 428, -2 – page 429, line 2]. *Let M be a T-submodule that gives rise to a permutation graph. Let D be any conformal model for G_c. Then the endpoints of chords of M can be partitioned into two sets A_1, A_2 such that each chord of M has one endpoint in A_1 and the other in A_2 and those endpoints in A_1 (resp., A_2) are consecutive in D.*

Theorem 6.6 [page 429, lines -15 to -11]. *Consider a neighborhood* (that is, prime, according to our terminology) *module $V(G_c)$. Let $V' = \{v_1, \ldots, v_r\}$ be a set of representatives from the maximal consistent submodules $M^1, \ldots, M^r$ in the consistent partition of $V(G_c)$. Then there is a unique conformal model for V'. Furthermore, this model is independent of the v_i selected (we shall refer to the subgraph induced on V' as the representative graph for $V(G)$).*

In fact, the two lemmas above establish Property (H3) for the consistent modules of G_c in the prime case. Notably, when G is co-bipartite, G_c is a permutation graph, and we can take the children of $V(G_c)$ in the modular decomposition tree of G_c as maximal consistent modules of G_c (recall that strong modules in the modular decomposition tree of a permutation graph are consistent in all its segment models). Hence, to determine the consistent modules of G_c Hsu focuses his attention on the children of $V(G_c)$ in the modular decomposition tree of G_c. Hsu rightly observes that they might be no longer consistent when G_c is not a permutation graph. In fact, assuming that $M_1, \ldots, M_k$ are the children of prime $V(G_c)$ in the modular decomposition tree of G_c, Hsu attempts to prove the following lemma:

Lemma 6.3 [Page 428, lines 11–12]. *If a maximal submodule M_i of $V(G_c)$ is not consistent, then it is a serial module.*

However, the above lemma is also false, as parallel children of $V(G)$ might also be non-consistent. This enables us to construct the graph claimed by Counterexample 1.



(a) A circular-arc graph G.

(b) Normalized model R of G.

(c) Associated chord model D of G_c.

(d) Graph G_c and its maximal non-trivial modules M_1, M_2, M_3, M_4.

Figure 7 Counterexample 1: $V(G_c)$ is a neighbourhood module, M_1, M_2, M_3, M_4 are all parallel children of $V(G_c)$ in the modular decomposition tree of G_c, and M_1 and M_4 are not consistent.

Counterexample 1. Consider a circular-arc graph G shown in Figure 7(a). Its normalized model R is shown in Figure 7(b), an associated chord model D in Figure 7(c), and the circle graph G_c associated to G and its partition into maximal non-trivial modules $M_1, \ldots, M_4$ in Figure 7(d). Note that $V(G_c)$ is a prime module in the modular decomposition tree of G_c,

and each M_i is a parallel child of $V(G)$. Note that the modules M_1 and M_4 are not consistent in D. One can also check that the model R and its reflection are the only two normalized models of G. Since both these models do not obey the description given by Hsu (it is claimed in [5] that only series children of $V(G)$ might be not consistent), the graph G proves the statement of Counterexample 1.

The graph G also shows that, assuming Hsu's partition into "consistent modules", Lemma 6.5 and Theorem 6.6 from [5] are also false.

4.2 Parallel case

In the case where $V(G_c)$ is a parallel module, the description given by Hsu is incomplete (and also not correct). In particular, Hsu constructs so-called "consistent module tree" for each component C of G_c basing on the consistent modules of the component C. As is written in [5] (see page 435, lines 27 to 29): *We shall first construct a consistent decomposition tree T for the subgraph $G_c[C]$ and then insert the vertices of K into conformal models of appropriate T-submodules. For any T-submodule M in C, let .....* and then a description of the tree T for the component C follows. Each component C of G_c is either a prime or a serial module in G_c. In the case when C is a prime component in G_c, the maximal consistent modules of C, as we have seen in the previous section, are not determined correctly by Hsu. Nowhere in [5] we found a description of the maximal consistent modules for serial components of G_c.

5 Oriented conformal models

The main difference between the approach used in [6] and Hsu's approach lies in the definition of the conformal models. In [6], a normalized circular-arc model ψ of G is transformed into an ***oriented*** *chord model* ϕ by converting every arc $\psi(v)$ for $v \in V(G)$ into an oriented chord $\phi(v)$ such that the chord $\phi(v)$ has the same endpoints as $\psi(v)$ and $\phi(v)$ is oriented so that it has the arc $\psi(v)$ on its left side – see Figure 8 for an illustration. Then, the following definition of conformal models is used.

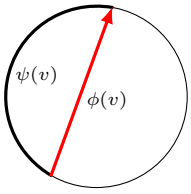


Figure 8 The transformation of the arc $\psi(v)$ into the oriented chord $\phi(v)$.

▶ **Definition 6.** *An oriented chord model ϕ of G_{ov} is* conformal to G *(shortly,* conformal*) if for every $v, u \in V$:*

- *$u \in L_v$ if and only if $\phi(u)$ lies on the left side of $\phi(v)$,*
- *$u \in R_v$ if and only if $\phi(u)$ lies on the right side of $\phi(v)$.*

See Figure 9 for an illustration. Clearly, there is an obvious correspondence between the normalized models of G and the oriented conformal models of G_c.

The lack of orientation in the chords of conformal models poses notable challenges in Hsu's work.

Firstly, Hsu divides all the triples (u, v, w) consisting of pairwise non-adjacent vertices in G_c into two categories:

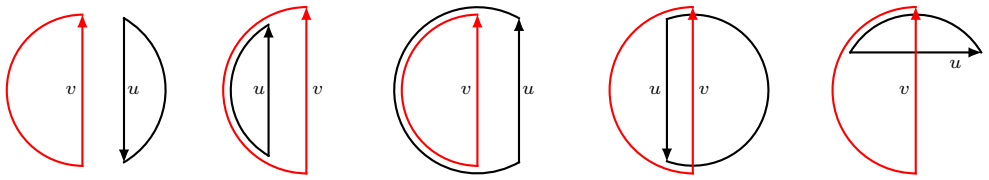


Figure 9 Relations between the arcs $\psi(v)$ and $\psi(u)$ and the corresponding oriented chords $\phi(v)$ and $\phi(u)$ for the cases: v is disjoint with u, v contains u, v is contained in u, v and u cover the circle, and v and u overlap, respectively.

- (u, v, w) is *in parallel*, written $u|v|w$, if the vertex v has the vertices u and w on its different sides (that is, either $u \in L_v$ and $w \in R_v$ or $w \in L_v$ and $u \in R_v$),
- (u, v, w) is *in series*, written $u - v - w$, if any vertex from $\{u, v, w\}$ has the remaining two vertices on the same side,

(see Section 5.1 of [5]). Then, Hsu is searching for chord models ϕ of G_c that satisfy the conditions:

- if $u|v|w$ then the chord $\phi(v)$ has $\phi(u)$ and $\phi(w)$ on its different sides,
- if $u - v - w$ then every chord from $\{\phi(u), \phi(v), \phi(w)\}$ has the remaining two chords on the same side.

Hsu showed that such chord models correspond to conformal models (see Section 5.2 in [5]). Consequently, Hsu aims to describe the set of transformations between the chord models of G_c that preserve the relationships between the triples of non-intersecting chords. By orienting the chords in the conformal models, we can concentrate on pairs of non-intersecting chords. Specifically, we seek transformations that maintain the relative (left/right) relationships between every pair of non-intersecting oriented chords. While this difference may seem minor, it significantly simplifies the task of identifying the set of operations that can transform one conformal model into another.

Secondly, reflecting two intersecting *non-oriented* chords a and b yields a pair of chords that intersect in exactly the same way. That is, reflection has no observable effect on two intersecting non-oriented chords; to detect a difference, additional chords must be involved. In contrast, when a and b are *oriented chords* corresponding to two overlapping arcs, reflection produces a different outcome. Indeed, if the head of b lies to the right of a before reflection, then after reflection, it lies to the left of a. See Figure 10 for an illustration. Understanding the role of reflection – specifically, which components of the conformal models can be reflected independently of others – is crucial for identifying all conformal models of G_c. In the case of oriented chords, the effects of the reflection can be at least easily described.

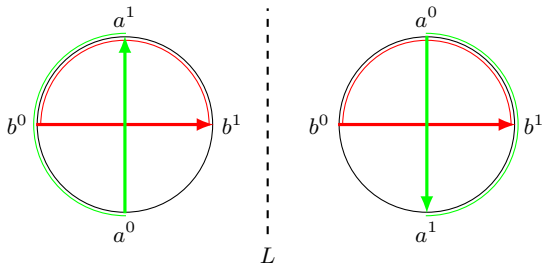


Figure 10 The result of the reflection of the arcs $a = a^0a^1$ and $b = b^0b^1$.

References

1 Andrew R. Curtis, Min Chih Lin, Ross M. McConnell, Yahav Nussbaum, Francisco J. Soulignac, Jeremy P. Spinrad, and Jayme Luiz Szwarcfiter. Isomorphism of graph classes related to the circular-ones property. *Discret. Math. Theor. Comput. Sci.*, 15(1):157–182, 2013. doi:10.46298/DMTCS.625.

2 Ben Dushnik and E. W. Miller. Partially ordered sets. *Amer. J. Math.*, 63:600–610, 1941. doi:10.2307/2371374.

3 Csaba P. Gabor, Kenneth J. Supowit, and Wen-Lian Hsu. Recognizing circle graphs in polynomial time. *J. ACM*, 36(3):435–473, 1989. doi:10.1145/65950.65951.

4 Tibor Gallai. Transitiv orientierbare Graphen. *Acta Math. Acad. Sci. Hung.*, 18(1–2):25–66, 1967.

5 Wen-Lian Hsu. O(M*N) algorithms for the recognition and isomorphism problems on circular-arc graphs. *SIAM J. Comput.*, 24(3):411–439, 1995. doi:10.1137/S0097539793260726.

6 Tomasz Krawczyk. On the structure of normalized models of circular-arc graphs - Hsu's approach revisited. *CoRR*, abs/2411.13374, 2024. doi:10.48550/arXiv.2411.13374.

7 Jeremy P. Spinrad. Circular-arc graphs with clique cover number two. *J. Comb. Theory B*, 44(3):300–306, 1988. doi:10.1016/0095-8956(88)90038-X.

Heuristics for Exact 1-Planarity Testing

Simon D. Fink
Algorithms and Complexity Group, TU Wien, Austria

Miriam Münch
Faculty of Computer Science and Mathematics, University of Passau, Germany

Matthias Pfretzschner
Faculty of Computer Science and Mathematics, University of Passau, Germany

Ignaz Rutter
Faculty of Computer Science and Mathematics, University of Passau, Germany

Abstract

Since many real-world graphs are nonplanar, the study of graphs that allow few crossings per edge has been an active subfield of graph theory in recent years. One of the most natural generalizations of planar graphs are the so-called 1-planar graphs that admit a drawing with at most one crossing per edge. Unfortunately, testing whether a graph is 1-planar is known to be NP-complete even for very restricted graph classes. On the positive side, Binucci, Didimo and Montecchiani [5] presented the first practical algorithm for testing 1-planarity based on an easy-to-implement backtracking strategy. We build on this idea and systematically explore the design choices of such algorithms and propose several new ingredients, such as different branching strategies and multiple filter criteria that allow us to reject certain branches in the search tree early on. We conduct an extensive experimental evaluation that evaluates the efficiency and effectiveness of these ingredients. Given a time limit of three hours per instance, our best configuration is able to solve more than 95% of the non-planar instances from the well-known `North` and `Rome` graphs with up to 50 vertices. Notably, the median running time for solved instances is well below 4 seconds.

2012 ACM Subject Classification Theory of computation → Backtracking; Mathematics of computing → Graph algorithms

Keywords and phrases 1-Planarity, Experiments, Backtracking

Digital Object Identifier 10.4230/LIPIcs.GD.2025.4

Funding Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 541433306.
Simon D. Fink: Simon D. Fink acknowledges support from Projects No. 10.47379/ICT22029 of the Vienna Science Foundation (WWTF) and No. 10.55776/Y1329 of the Austrian Science Fund (FWF).

1 Introduction

It has been established early on in the history of Graph Drawing that the number of crossings has a major influence on the readability of drawings [37, 38, 39]. This reinforces the continued interest in the extensively studied *crossing minimization problem*, which asks for the minimum number of crossings over all drawings of a given graph. It was shown as early as 1983 that the problem is NP-complete [20]. Due to its foundational nature, it has fueled a large number of results that address combinatorial and algorithmic aspects of crossing numbers. We refer to the survey of Schaefer for an overview of this extensive field [44]. Due to the practical importance of the problem, these developments have early on been paralleled by more applied works that are concerned with solving the crossing minimization problem in practice. Today, there is a large amount of literature that deals with computing the crossing number either exactly in exponential time, based on formulations as integer linear programs [8, 13, 33], or heuristically in polynomial time [11, 12, 22]. Heuristics often work by efficiently and


© Simon D. Fink, Miriam Münch, Matthias Pfretzschner, and Ignaz Rutter;
licensed under Creative Commons License CC-BY 4.0



optimally inserting vertices or edges into a drawing, allowing to introduce crossings only between pairs of edges where at least one of the edges is currently being inserted; a strategy that has also proved to be successful for straight-line drawings [6, 40, 41, 42].

A more recent development is based on the hypothesis that in addition to the number of crossings, a major influence on the readability of drawings comes from the distribution of the crossings within the drawing. This has lead to the definition of so-called *beyond-planar graphs* that admit drawings whose crossings avoid certain patterns that are considered to impede readability. Prominent examples of beyond-planarity concepts are k-planarity, k-quasi-planarity, fan-planarity, gap-planarity, RAC-planarity, min-k-planarity as well as corresponding outer and upward variants; we refer to [26] for a survey. One of the most widely known beyond-planarity concepts is *1-planarity*, where a graph is called 1-planar if it can be drawn in the plane such that each edge has at most one crossing. Despite the apparent simplicity, 1-planar graphs form a rich class that has spurred a substantial amount of research. From a combinatorial perspective, questions about density, coloring and structural decompositions have been deeply investigated; we refer to [27, 45] for surveys.

From an algorithmic perspective, it is well known that testing 1-planarity is NP-complete [21, 28], even for graphs of bounded bandwidth, pathwidth or treewidth [4], as well as for graphs that can be made planar by the removal of a single edge [9] or graphs with a fixed rotation system [3]. Polynomial-time recognition algorithms are only known for restricted subclasses such as outer 1-planar graphs [2, 25], maximal 1-planar graphs with a fixed rotation system [19], 1-planar maximal graphs of pathwidth w [31], and optimal 1-planar graphs [7]. Recent years have seen a plethora of fixed-parameter tractability (FPT) results on testing 1-planarity. In particular, Bannister, Cabello and Eppstein [4] showed that the problem is FPT with respect to the vertex cover number, the tree-depth, and the cyclomatic number. Various recent results, see e.g. [16, 23, 24, 32], imply that 1-planarity is FPT with respect to the 1-planar crossing number, i.e., the minimum number of crossings over all 1-planar drawings of a graph. The mentioned results are however only of theoretical interest as they rely on deep theoretical results such as Courcelle's Theorem [15] or testing embeddability of graphs in 2-complexes [17].

A key motivation that initiated the study of beyond-planarity was the observation that while many real world-graphs are non-planar, they are usually sparse and close to planarity in the sense described above, namely that they have drawings whose crossings are well-scattered [26]. Correspondingly, the problem of recognizing beyond-planar graphs and in particular testing 1-planarity has a strong practical motivation. Despite this and in stark contrast to the problem of crossing optimization, where the theoretical developments have been accompanied by corresponding practical results, there is a distinct scarcity of practical algorithms in the area of beyond-planarity. In fact, we are only aware of two attempts.

Angelini, Bekos, Kaufmann and Schneck [1] present an algorithm that is based on enumerating all topological drawings of a graph incrementally by iteratively inserting the edges. This is useful, e.g., to devise small counterexamples or to confirm conjectures on small graphs with less than 15 vertices, but it is hardly useful for practical applications. Binucci, Didimo and Montecchiani [5] present an easy-to-implement algorithm, called `1PlanarTester`, for recognizing 1-planar graphs that is based on backtracking and is quite effective for small graphs. As a graph is 1-planar if and only if all its biconnected components are 1-planar, they process the biconnected components independently. For each biconnected component, they fix a global ordering σ on the set of all pairs of non-adjacent edges, consider each such *crossing candidate* pair $\{e, f\}$ in the order of σ and branch on the binary decision of whether e and f should cross. This is repeated until a 1-planar drawing is found or the

whole search tree has been explored and the instance is rejected. Branches that cannot lead to a solution, e.g., when an edge receives more than one crossing, or there is a non-planar subgraph whose edges may not receive further crossings, are excluded from the backtracking. The authors extend this basic algorithm by combining it with several *filter criteria* that exclude additional branches, e.g., based on the addition of "kite" edges around crossings and density bounds for planar and 1-planar graphs. The algorithm can handle graphs with up to 30 vertices in a reasonable amount of time. However, on graphs with up to 50 vertices the algorithm often either gives a positive answer in less than 0.2 seconds or does not give an answer within a three-hour time limit. This emphasizes the crucial role of the order σ; either a lucky guess of helpful crossings early on paves the ways towards a solution, or the algorithm more or less explores the entire search tree.

Contribution and Outline. We build on the ideas of Binucci et al. [5] and systematically explore different design choices within a backtracking framework. To this end we propose new branching strategies aimed at limiting the size of the tree that has to be explored and novel filter criteria that enable the earlier rejection of branches in the search tree. We conduct an extensive experimental evaluation that examines the efficiency and effectiveness of these techniques. Given a time limit of three hours per instance, our best configuration significantly outperforms the `1PlanarTester` and is able to solve more than 95% of the non-planar instances from the well-known `North` and `Rome` graphs with up to 50 vertices. On the whole data set, our best configuration solves 88% of all non-planar `North` graphs and 61% of all non-planar `Rome` graphs, which contain instances with up to 110 vertices.

We give basic definitions in Section 2. In Section 3 we describe our underlying backtracking strategy and the corresponding search tree followed by a description of our traversal strategy. Section 4 is then devoted to the presentation of different branching strategies and in Section 5 we describe multiple filter criteria. In Section 6 we present the results of our experimental evaluation. We conclude with a brief summary in Section 7.

We remark that Pupyrev simultaneously and independently designed and implemented an algorithm for testing 1-planarity based on a reduction to a SAT-instance [36].

2 Preliminaries

We assume familiarity with basic graph terminology. Throughout this paper, all graphs are *simple*, i.e., they have neither parallel edges nor loops. We use K_n to denote the complete graph on n vertices and $K_{n,m}$ to denote the complete bipartite graph whose bipartition contains n and m vertices, respectively.

A *drawing* Γ of a graph $G = (V, E)$ maps every vertex $v \in V$ to a point $\Gamma(v)$ in the plane and every edge $uv \in E$ to a Jordan arc with endpoints $\Gamma(u), \Gamma(v)$ that does not pass through any $\Gamma(w)$ for $w \in V \setminus \{u, v\}$. For simplicity, we identify vertices and edges with their images in a drawing. A *crossing* is a common interior point of two edges. A drawing Γ of a graph G is *1-planar*, if every edge is crossed at most once in Γ. The *planarization* of a drawing Γ is the graph we obtain from Γ by replacing every crossing in Γ with a dummy vertex. A drawing subdivides the plane into connected regions called *faces*, one of which is unbounded and is called *outer face*. An *embedding* of a graph G is the equivalence class of those drawings of G that share the same face structure and cyclic order of the incident edges around each vertex. A 1-*planar embedding* is an embedding induced by a 1-planar drawing and a 1-*plane graph* is a graph together with a fixed 1-planar embedding. A *kite* is a 1-plane graph isomorphic to K_4 whose outer face is bounded by a cycle of four vertices and four uncrossed edges, called *kite edges*, while the remaining two edges cross each other; see Figure 1.

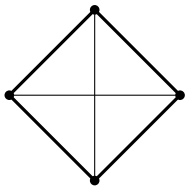


Figure 1 A kite with kite edges in bold.

It is well known that a 1-planar graph on n vertices has at most $4n - 8$ edges [35]. We call a graph with more than $4n - 8$ edges *trivially non-1-planar*. A graph is *biconnected*, if it is connected and remains connected after the removal of any vertex. Recall that a graph is non-planar if and only if it contains a subgraph that is isomorphic to a subdivision of $K_{3,3}$ or K_5 as a subgraph [29]. We call such a subgraph a *Kuratowski-subdivision*.

3 The Backtracking Procedure

In this section we formalize the backtracking strategy we use in the rest of this paper. It is based on a search tree similar to the one of Binucci et al. [5] described in the introduction, whose nodes correspond to partial solutions. Recall that the algorithm of Binucci et al. constructs the search tree based on a global order σ of the crossing candidates. Thus, in their setting, a partial solution corresponds to a bitstring that describes the choices made on a prefix of the order σ. As mentioned in the introduction, the performance seems to crucially depend on σ. We conjecture that, rather than fixing a global order σ on the crossing candidates, it may be beneficial to choose the next crossing candidate to branch on according to criteria that take into account the current partial solution and thus allow different subtrees to explore the same crossing candidate at different depths of the search tree. This may enable the search to focus on critical parts of a partial solution and has the potential to reject infeasible partial solutions more quickly. For this reason, we choose a different representation of partial solutions that encodes the current state more explicitly.

As a graph is 1-planar if and only if each of its biconnected components is 1-planar, we assume in the following that our input graph $G = (V, E)$ is biconnected. Before we describe our backtracking strategy, we introduce some useful notation. For a crossing candidate $\{e, f\}$ with $e = uv$, $f = wy$ let $G_{\{e,f\}}$ denote the graph we obtain by inserting a dummy vertex x subdividing e and f, i.e., $G_{\{e,f\}} = (V \cup \{x\}, (E \setminus \{e, f\}) \cup \{ux, vx, wx, yx\})$. We say that the edges ux, vx and wx, yx *stem from* e and f, respectively. For a set C of crossing candidates such that each edge in E is contained in at most one pair of C, we denote by G_C the graph we obtain from G by inserting a dummy vertex as described above for every pair in C.

Let $\bar{E} \subseteq \binom{E}{2}$ be the set containing all crossing candidates $\{e, f\}$ consisting of non-adjacent edges e and f. We encode a partial solution for G as a pair (C, P) with $C, P \subseteq \bar{E}$. The intended meaning is that we have chosen the crossing candidates in C and we may still choose among the crossing candidates in P. Since each edge may receive at most one crossing, we require that for each edge $e \in E$ that appears in a pair $\{e, f\} \in C$, the pair $\{e, f\}$ is the only pair in $P \cup C$ that contains it. We call an edge of G_C *free* if it is contained in a pair in P and we call it *saturated* otherwise. Observe that G is 1-planar if and only if there exists a partial solution (C, P) where G_C is planar. A partial solution (C, P) is *extendible* if there exists a subset $C' \subseteq P$ such that $G_{C \cup C'}$ is planar.

The backtracking procedure incrementally constructs a search tree T starting from the partial solution $(\emptyset, \bar{E})$. We process a node of T representing a partial solution (C, P) as follows. First, if G_C is planar, we return the corresponding 1-planar embedding of G. Second,

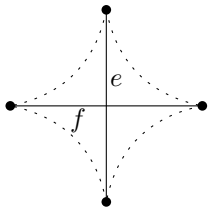


Figure 2 The dotted degree-2-paths can always be redrawn without any crossing.

if G_C is not planar but $P = \emptyset$, we backtrack, since (C, P) does not extend to a 1-planar drawing. If neither of these applies, we use a *branching strategy* to compute a finite set $\mathcal{C}$ of child partial solutions (C', P') with $C \subseteq C' \subseteq C \cup P$ and $P' \subsetneq P$. The chosen set $\mathcal{C}$ has to be *exhaustive* in the sense that the partial solution (C, P) is extendible if and only if the same holds for at least one child partial solution in $\mathcal{C}$.

We note that this basic framework leaves several degrees of freedom. First, the choice of the branching strategy can have a significant impact on the size of the search tree. Second, in addition to that, the order in which the nodes of T are processed can have a major impact on the time it takes to find a solution. We discuss different options in Section 4.

Third, a major issue stems from the fact that for no-instances the entire tree T needs to be traversed. If we identify that a certain partial solution (C, P) is not extendible, its subtree contains no solution and we can immediately backtrack. Hence, a necessary criterion for extendibility of (C, P) can be used to prune the search tree whenever the criterion is not satisfied. We call such a criterion a *filter*. An example of such filter is excluding nodes (C, P) where G_C is trivially non-1-planar. We propose additional filters in Section 5.

Another optimization already described by Binucci et al. [5] exploits the fact that kite edges can always be redrawn such that they do not cross any other edge. In particular, when adding a new crossing $\{e, f\}$ to the currently considered partial solution, they insert all edges connecting an endpoint of e to an endpoint of f that are not already present in G and make all these kite edges *uncrossable*, which in our setting corresponds to removing from P all pairs that contain such a kite edge. We note that this can be slightly strengthened by making all edges uncrossable that are contained in a path from an endpoint of e to an endpoint of f whose internal vertices are all of degree 2, as such paths can also always be redrawn arbitrarily closely along e and f without any crossings; see Figure 2. The described optimizations help in keeping the search tree small as the number of crossings that have to be considered decreases, and violations of necessary conditions on the extendability to a 1-planar embedding are found earlier during the execution. We thus include them in each of our algorithms.

4 Branching and Traversal Strategies

In this section, we propose different branching strategies for choosing the children of a partial solution (C, P) and discuss different options for the traversal of the resulting search tree. We discuss branching strategies in Section 4.1 and traversal strategies in Section 4.2.

4.1 Branching Strategies

Let (C, P) be a partial solution and assume that G_C is non-planar and $P \neq \emptyset$.

Binary Branching. A basic branching strategy consists of choosing a crossing candidate $x \in P$ and creating children $(C, P \setminus \{x\})$ and $(C \cup \{x\}, P')$, where P' is obtained from P by removing all crossing candidates that contain an edge from x. We refer to such a strategy as *binary branching* and we consider three variants of it. The first is the one also employed by `1PlanarTester` of Binucci et al. [5], where we initially choose a (random) global order σ of the crossing candidates and x is always chosen as the smallest element of P with respect to this order. We refer to this strategy as `Sequential Branching`. For `Random Branching`, we choose x uniformly at random from P and for `KuratowskiFrequency-`k `Branching`, we extract k Kuratowski subdivisions from the non-planar graph G_C and choose x such that the corresponding pair of edges is together contained in a maximum number of these Kuratowski subdivisions. We remark that this can be done in linear time using an algorithm by Chimani, Mutzel and Schmidt [14].

Exhaustive Set Branching. The other major branching strategy we explore is *exhaustive set branching*, where we use an ordered set $P' \subseteq P$ of edge pairs with the property that every 1-planar embedding of G containing the crossings in C also contains at least one crossing from P'. Given such a set $P' = \{p_1, \ldots, p_k\}$, we produce k children (C_i, P_i), where $C_i = C \cup \{p_i\}$ and P_i is obtained from $P \setminus \{p_1, \ldots, p_i\}$ by removing all crossing candidates that contain an edge of p_i. That is, in the ith child, the crossing candidate p_i is chosen and the crossing candidates $p_1, \ldots, p_{i-1}$ are rejected.

Our choices of P' and its ordering described below are guided by which edge pairs are more likely to correspond to "helpful" crossings and thus should be tried with higher priority. In all cases, we use Kuratowski subdivisions of G_C to guide us towards the set P' as, for any Kuratowski subdivision H of G_C, at least one pair of non-adjacent edges that do not belong to the same subdivision path need to be crossed. We call this set $P(H)$.

In its simplest form, called `KuratowksiSingle Branching`, we extract from G_C a single Kuratowski subdivision H and branch on the set $P(H) \cap P$ with an arbitrary order. Kuratowski subdivisions that are small or have few free edges produce fewer children, which can significantly narrow the search space. Strategy `KuratowskiMulti-`k `Branching` exploits this by extracting a set $\mathcal{H}$ of k Kuratowski subdivisions and choosing the subdivision $H \in \mathcal{H}$ that minimizes $P(H) \cap P$ to branch on. To increase the likelihood of trying helpful crossings early, we further order the crossing candidates in $P(H) \cap P$ by the number of Kuratowski subdivisions in $\mathcal{H}$ that contain them.

4.2 Traversal Strategies

The choice of a branching strategy determines the search tree T. However, there are still different options for traversing T. The algorithm `1PlanarTester` of Binucci et al. [5] employs a depth-first search (DFS) to traverse the search tree. This has the effect that a few poor initial choices may necessitate the exploration of a large subtree that does not contain a solution. Especially for larger instances, the algorithm may not recover from this in a timely manner as it has to explore the entire subtree.

An obvious alternative to this is the use of a breadth-first search (BFS), which prioritizes partial solutions with few crossings and may thus even allow to find a 1-planar embedding with the minimum number of crossings. However, due to the large size of T, the corresponding memory requirement is prohibitive as essentially the entire tree needs to be kept in memory.

We thus propose a hybrid approach. We maintain a queue Q with up to k distinct *threads* that correspond to independent parallel depth-first searches. In each step we extract the next thread t from the queue and execute one step of the corresponding DFS. When processing a

node (C, T), while Q contains fewer than k threads, instead of moving to a child, we rather insert a new thread into Q that starts a depth-first search at this child. When Q reaches the maximum of k threads, the remaining unvisited children are pushed to the stack of the depth-first search corresponding to t.

In comparison to traversing T by a single DFS, in case of a yes-instance, this may prevent us from fully exploring a very deep unhelpful branch and may allow to find potential solutions earlier during the execution. We note that, for no-instances the whole tree needs to be explored and the choice of the traversal strategy thus does not affect the running time.

5 Filter Criteria

In this section, we describe several filters to determine that a partial solution (C, P) is not extendible and, therefore, allows us to ignore its entire subtree. This has significant potential to speed up the solution speed for both 1-planar and non-1-planar instances.

We first describe the filters that were already used by Binucci et al. [5] and that we employ in all our strategies. We phrase each of them as a necessary condition, whose violation allows to cut the corresponding subtree. The first, called `Planar Structure`, requires that the subgraph of G_C induced by the saturated edges is planar. To strengthen this, we also include the kite edges and the extended kite edges as described at the end of Section 3. The filter `Edge Density` rejects instances where G_C is trivially non-1-planar after adding potentially missing kite edges. We propose three further types of filters.

Partial Planarity. Let (C, P) be a node and let H be the subgraph of G_C that consists of the saturated edges. Partial planarity filters are based on the fact that if (C, P) is extendible, then G_C admits a drawing where H is crossing-free (but all other edges may cross arbitrarily). The corresponding decision problem is known as Partial Planarity. Da Lozzo and Rutter [30] gave a linear-time algorithm, which is however very involved. Therefore, we instead implement the slower but simpler Hanani-Tutte style algorithm of Schaefer [43]. Due to the high running time of $O(n^6)$, we initially run cheaper tests of weaker necessary properties as described below. For each variant, we assume that all previous variants are used as well.

- `PP1:` A necessary condition for partial planarity is that for each free edge e, the graph $H+e$ is planar. We thus test this for each edge and reject if this is not the case.
- `PP2:` Let $\{e, f\}$ be a pair of free edges in G. Assuming PP1 has not rejected, we know that $H + e$ and $H + f$ are planar. If $H + \{e, f\}$ is non-planar, we know that e must cross f. In this case, if $\{e, f\} \in P$, we can include this crossing in C and otherwise we can reject the partial solution. We check for each pair of free edges in G whether it is forced in this way.
- `PP3:` Let B be a connected component of $G_C - H$ together with all edges that connect it to H (along with their endpoints in H). We call such a subgraph a *bridge* of G_C. Now in a potential 1-planar embedding of G with all crossings from C, each bridge must lie entirely within one face of H. In other words, for the extendability of the partial solution, it is necessary that the graph consisting of H and the bridge B, where B is contracted into a single vertex, is planar.
- `PP4:` Finally, if none of the cheaper tests rejects the partial solution, we run the $O(n^6)$ partial planarity algorithm by Schaefer [43].
  Due to the high running time of `PP4`, we also propose a variant `PP4C`p, where we only run the test with probability $\frac{1}{p}$. In that case, if we detect that G_C is not partially planar,

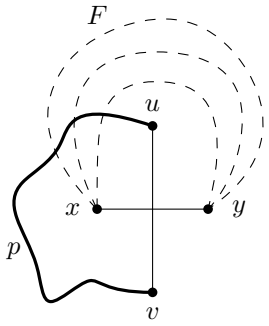


Figure 3 If $e = uv$ crosses $f = xy$, then in any 1-planar drawing, all pairwise edge-disjoint paths between x, y in F (dashed) cross the path p between u and v (bold).

we also run the test on ancestors in the search tree that have not been tested previously in order to find the highest ancestor x whose corresponding planarization is also not partially planar. We can then prune the entire subtree rooted at x. In this way, a single invocation of the test can prune multiple nodes from the search tree.

Separating Cycles (`SC`). Let (C, P) be a configuration in the search tree T and let $e = uv, f = xy$ be two edges in G that cross each other in the corresponding partial solution; i.e, $\{e, f\} \in C$. Let p be a uv-path in $G - \{e, f\}$ that contains the minimum number of free edges. Let further F be a maximum set of pairwise edge-disjoint paths between x and y such that no path in F shares a vertex with p; see Figure 3 for an illustration. Since e and f cross, the endpoints x and y of f lie on different sides of the cycle K formed by p and e. Thus each path in F needs to cross K. It is thus necessary that each path of F contains an edge that still admits a crossing with an edge of K and that K contains at least $|F|$ free edges.

In our implementation, if the test succeeds for a path p we temporarily remove it from the graph and repeat the test until there is no longer a path that connects x to y. We execute this test for all pairs $\{e, f\} \in C$.

6 Experimental Evaluation

In this section, we experimentally evaluate our backtracking procedure. A concrete instance of our backtracking procedure, called a *configuration* is obtained by choosing a branching strategy and a traversal strategy from Section 4 along with a subset of the filters from Section 5. As some of the strategies have additional parameters, this yields a relatively large number of possible configurations. To address this, we first perform preliminary experiments on a smaller test set and with a short time limit to assess the impact of the individual strategies and filter criteria and to ultimately determine the configuration with the best performance. Afterwards, we compare the performance of this winning configuration with that of the algorithm `1PlanarTester` of Binucci et al. [5] by running it on an extensive test set in a way that closely mimics their experimental setup. As an additional reference point, we also implemented a simple ILP-based algorithm that builds on the ILP formulation of Buchheim et al. [8] for the Exact Crossing Number Problem.

In the following, we first describe the ILP formulation followed by a description of our test data and the experimental setup. Finally we report the results of our evaluation.

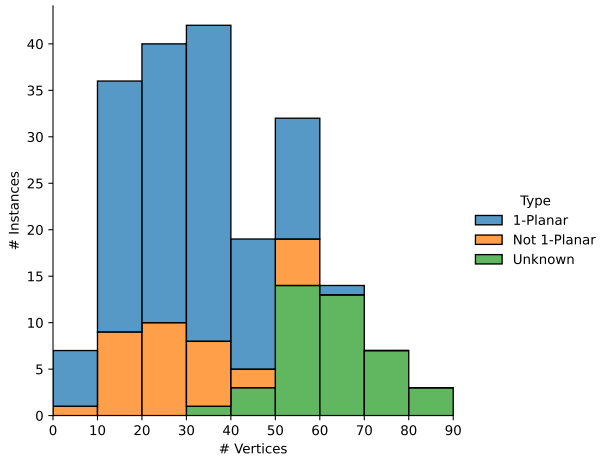


Figure 4 The distribution of instance sizes for the data set `Bicomps-NR-Small`.

6.1 ILP Formulation

In this section we describe our ILP-based algorithm which builds on the ILP formulation of Buchheim et al. [8] for the Exact Crossing Number Problem. Given the input graph G, for each candidate crossing $\{e, f\}$ of edges of G, we introduce a binary variable that indicates whether e and f cross in a solution and add constraints that ensure that every edge is contained in at most one crossing. We then repeat the following steps.

1. If the model is infeasible, reject G as not 1-planar.
2. Otherwise, consider an arbitrary solution of the model and let G' denote the graph obtained by replacing each edge pair that is crossed in the solution with a crossing vertex.
   a. If G' is planar, then report G as 1-planar.
   b. Otherwise, consider a Kuratowski-subdivision K of G'. Add a constraint to the model that prohibits K, i.e., ensure that (1) an edge pair is crossed such that K is removed or (2) a crossed edge pair whose crossing vertex is contained in K is "uncrossed". Then continue with Step 1.

Note that our strategies and filters from the preceding sections cannot easily be applied to the ILP-formulation.

6.2 Test Data

To make our results comparable to the results by Binucci et al. [5], we also work with the `North` and `Rome` graphs [34], two well-established test sets of graphs that stem from real-world applications. Since planar graphs are trivially 1-planar, we restrict ourselves to the non-planar instances. We let `NR` denote the test set consisting of all non-planar `North` and `Rome` graphs.

Since a graph is 1-planar if and only if each of its biconnected components is 1-planar, we believe that benchmarks including the input size are more representative if one considers each biconnected component individually. Therefore, we further decompose the graphs from `NR` into their non-planar biconnected components for most of our tests. For our initial evaluation of the different configurations of our backtracking procedure, we use a smaller data set `Bicomps-NR-Small` that consists of 100 randomly chosen non-planar biconnected components from the `North` graphs and 100 randomly chosen non-planar biconnected com-

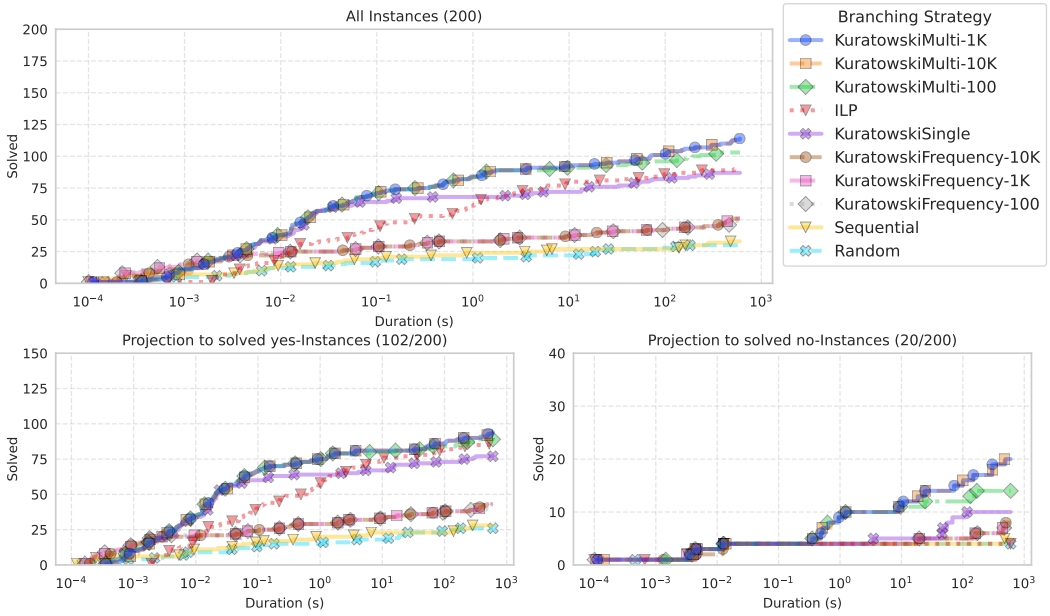


Figure 5 The number of solved instances over time on the 200 graphs of `Bicomps-NR-Small` for each branching strategy from Section 4.1. The suffix -x for `KuratowskiMulti` and `KuratowskiFrequency` denotes the number of extracted Kuratowski subdivisions.

ponents from the `Rome` graphs. Figure 4 shows the distribution of the instances in this data set. To eventually evaluate the best configuration, we also use the data sets `Bicomps-N` and `Bicomps-R` containing all non-planar biconnected components from the `North` and `Rome` graphs, respectively.

6.3 Experimental Setup

Our implementation is written in C++ and uses the Open Graph Drawing Framework (OGDF) [10] for graph representation and algorithms. The code was compiled using GCC version 12.2.0 and optimization `-O3`. Each experiment was run on a single core of an Intel Xeon E5-2690v2 CPU (3.0 GHz) with a memory limit of 10 GiB and running Linux Kernel version 6.1.0-37. We use Gurobi version 12.0.2 as a solver for our reference ILP.

6.4 Evaluation of Backtracking Strategies

In this section, we run preliminary experiments on the data set `Bicomps-NR-Small` with a time limit of ten minutes to determine the best configuration for our backtracking procedure. Each configuration includes kite edges and the basic filters for edge density and non-planarity of the saturated subgraph, as these optimizations are cheap in terms of running time.

Branching Strategies. We first evaluate the different strategies proposed in Section 4.1 for choosing the crossing candidates to branch on in every node of the search tree with a DFS as traversal strategy and only the basic filters; see Figure 5 for the results. Unsurprisingly, the two binary branching strategies, which do not consider the structure of the graph in any way (`Sequential` and `Random`) perform the worst. In comparison, the binary branching strategy `KuratowskiFrequency`, which prioritizes edge pairs that appear in many Kuratowski

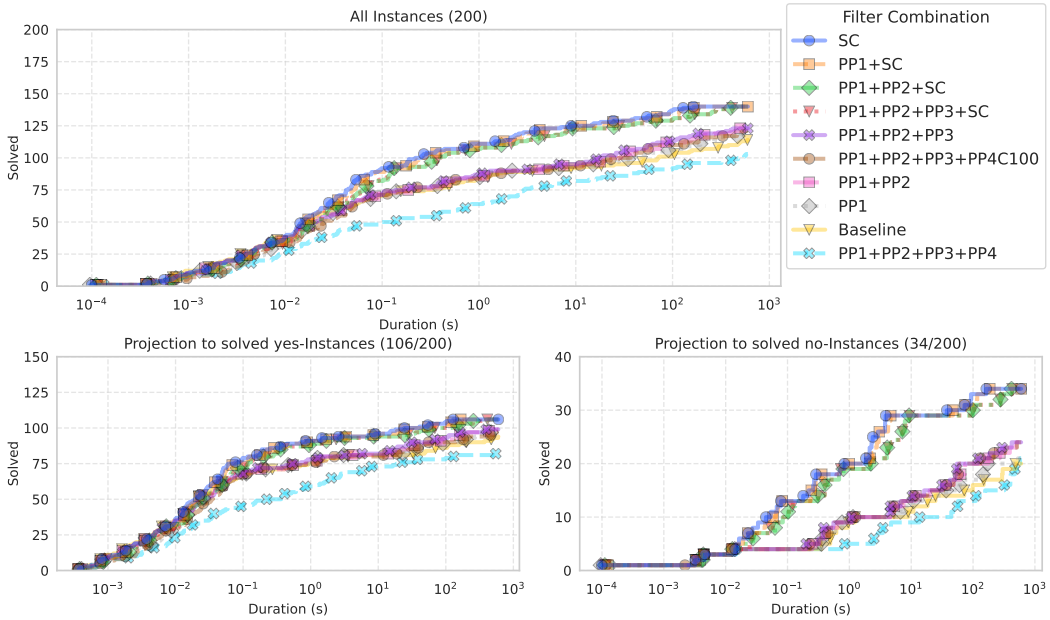


Figure 6 The number of solved instances over time on the 200 graphs of `Bicomps-NR-Small` for different combinations of filters from Section 5.

subdivisions, solves roughly 50% more instances than `Sequential`. Unlike `Sequential` and `Random`, `KuratowskiFrequency` only considers edge pairs that belong to a Kuratowski subdivision, which also improves its performance for no-instances; see Figure 5. Considering more than 100 Kuratowski subdivisions does not make a notable difference for this strategy.

The two exhaustive set branching strategies (`KuratowskiSingle` and `KuratowskiMulti`) clearly outperform the binary branching strategies. This is due to the fact that the search space significantly shrinks, as the branch where no edge pair of the chosen Kuratowski subdivision is crossed can be omitted. Interestingly, the ILP formulation also performs significantly better than the binary branching strategies and is on par with `KuratowskiSingle`. It performs better than `KuratowskiSingle` on yes-instances, but is only able to recognize no-instances that are trivially non-1-planar. The clear overall winner is the strategy `KuratowskiMulti`, where we consider multiple Kuratowski subdivisions and branch over the one with the lowest number of free edge pairs, prioritizing edge pairs that occur in many Kuratowski subdivisions. While `KuratowskiMulti` clearly benefits from extracting 1000 Kuratowski subdivions instead of 100, increasing this number to 10000 makes no notable difference. This could be at least partly due to the fact that the graphs that have such a high number of Kuratowski subdivisions are too large to be solved within the ten minute timelimit. However, since extracting more Kuratowski subdivisions comes at a higher cost of running time, we choose `KuratowskiMulti-1K` as the winning branching strategy for the subsequent experiments.

Filter Criteria. Next, we consider the filter criteria introduced in Section 5. As a baseline, we use the winning branching strategy `KuratowskiMulti-1K` from above. Figure 6 illustrates the solved instances over time for different (combinations of) filters. Moreover, Table 1 shows the number of solved instances and the corresponding *success rate* of each filter, i.e., the number of nodes in the search tree that were rejected by the filter divided by the total number of its invocations. The simplest partial planarity filter (`PP1`) alone rejects almost

Table 1 The number of solved instances and filter success rates for different combinations of the filters in Section 5 on the data set `Bicomps-NR-Small`. The left-to-right order of the columns indicates the order in which the filters are invoked in the implementation. Note that the ratio greater than 1 for `PP4C100` stems from the pruning of ancestors; see Section 5.

Filter Combination	Solved	Filter Success Rate				
		PP1	PP2	PP3	PP4	SC
Baseline	114	-	-	-	-	-
PP1	120	0.58	-	-	-	-
PP1+PP2	123	0.37	0.38	-	-	-
PP1+PP2+PP3	123	0.27	0.35	0.51	-	-
PP1+PP2+PP3+PP4	103	0.2	0.27	0.41	0.26	-
PP1+PP2+PP3+PP4C100	121	0.26	0.33	0.48	1.29	-
SC	140	-	-	-	-	0.9
PP1+SC	140	0.08	-	-	-	0.89
PP1+PP2+SC	140	0.08	0.14	-	-	0.87
PP1+PP2+PP3+SC	140	0.08	0.14	0.2	-	0.84

60% of all nodes in the search tree it is invoked on and this already increases the number of solved instances by roughly 5% compared to the baseline. Additionally enabling `PP2` and `PP3` only slightly improves the number of solved instances. Unsurprisingly, due to the $O(n^6)$ running time of `PP4`, also enabling this filter is detrimental for the algorithm due to its high execution time. Running the filter only 1% of the time (`PP1+PP2+PP3+PP4C100` in Table 1) raises the overall success rate of the filter over 100%, since we retrospectively test ancestors of nodes where the filter was successful and prune the search tree accordingly (see Section 5). However, it is overall still better to not enable this filter. Instead, using the filter `SC` yields a significant improvement and allows the backtracking procedure to solve roughly 23% more instances than the baseline overall, independent from what other filters are enabled.

In order to evaluate the filter combinations in more detail, we additionally use *performance profiles* [18]. For a set of algorithms and a specific performance metric (such as computing time), a corresponding performance profile shows for each algorithm A for what percentage of instances the performance of A lies within a certain distance of the best performance among all algorithms. In Figure 7, we first consider the computing time as the performance metric. Clearly, the combinations including the filter `SC` significantly outperform all combinations where `SC` is disabled. In fact, the algorithm has the best performance if only the filter SC is enabled: it is the fastest combination for over 50% of the solved instances and its running time differs at most by a factor of 1.5 from the fastest combination for around 90% of the instances that are solved by any of the algorithms. This indicates that `SC` dominates the other filters, i.e., it most likely rejects many configurations in the search tree earlier than the other filters would. This theory is supported by the fact that the success rate of the other filters shrinks significantly if combined with `SC`, despite the fact that `SC` is invoked last; see Table 1. Overall, `SC` alone rejects around 90% of all search tree nodes it is invoked on.

Altogether, this suggests that enabling only the filter `SC` is the winning configuration. However, it should be noted that this evaluation is potentially unfair towards the more costly filters such as `PP4`. Our implementation has a rather high time complexity of $O(n^6)$, which likely prohibits the algorithm from exploring a significant number of nodes of the search tree within the time limit. Since, theoretically, the filter `PP4` could be implemented with linear running time [30], it would be premature to reject `PP4`.

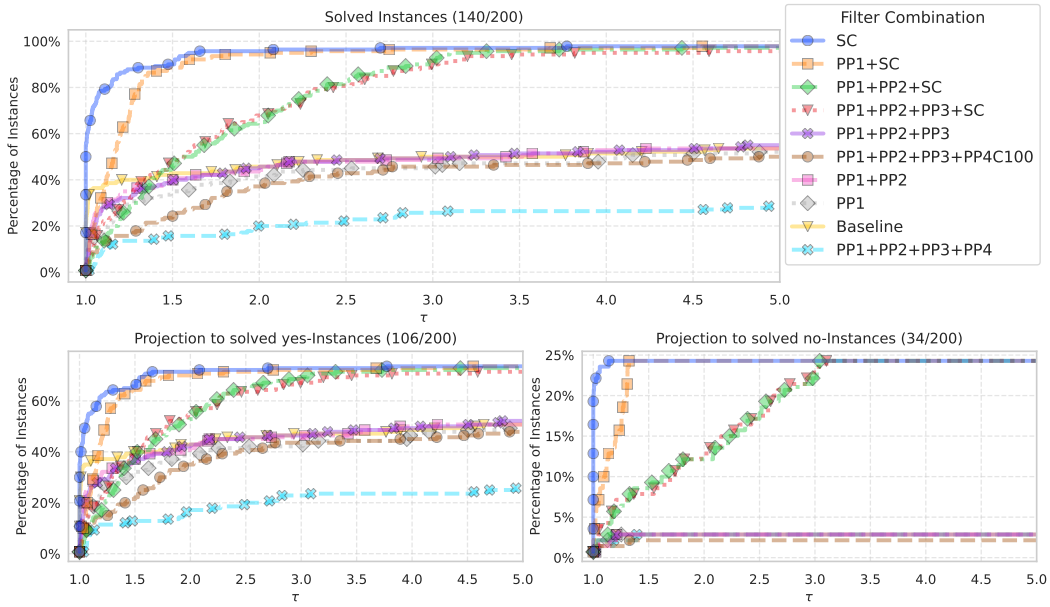


Figure 7 The performance profiles [18] for different filter combinations with the computing time as the performance metric. For each filter combination A, the plot shows the percentage of instances where the computing time of A is at most by a factor of τ slower than the fastest time among all combinations.

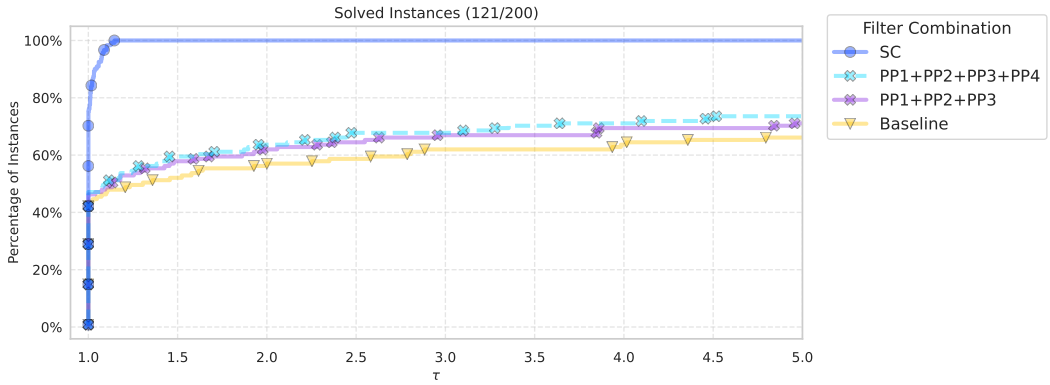


Figure 8 The performance profiles [18] for different filter combinations with the number of processed nodes (instead of the running time) as the performance metric. For each filter combination A, the plot shows the percentage of instances where the number of processed nodes of A is at most by a factor of τ larger than the lowest number of processed nodes among all combinations. Instead of a time limit, the algorithms were given a limit of 5000 nodes in the search tree that can be processed for each instance.

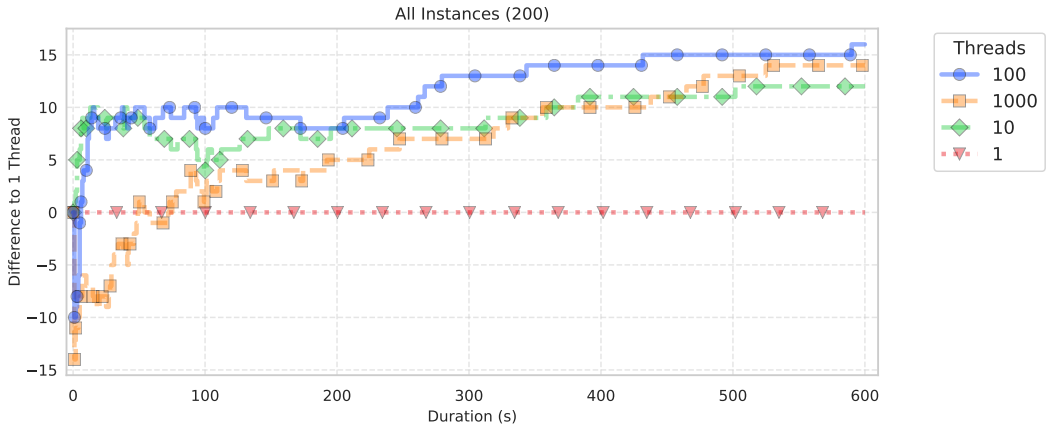


Figure 9 The performance of different thread counts for branching strategy `KuratowskiMulti-1K` with filter `SC` on the data set `Bicomps-NR-Small`. The plot shows the absolute difference in the number of solved instances over time to the baseline using only a single thread.

To better assess the potential of the filters, independent of their associated costs in terms of the running time of our implementation, we additionally consider performance profiles in Figure 8, where we consider as the performance metric the number of processed nodes instead of the running time. Instead of a time limit, the implementations are given a limit of 5000 nodes in the search tree that can be processed. It is not surprising that, if running time is disregarded, it is always better to use as many filters as possible. However, Figure 8 clearly shows that, even in this case, `PP4` only offers a marginal improvement over the simpler variants `PP1`-`PP3` and `SC` alone performs significantly better than all other filters. This indicates that even a linear-time implementation of `PP4` would most likely not yield a meaningful improvement of the backtracking procedure. Moreover, Figure 8 shows that the configuration using only the filter `SC` processes at most 15% more nodes in the search tree than the other combinations. We thus exclusively use `SC` in our winning configuration.

Traversal Strategies. Finally, we investigate the effect of threads (see Section 4.2) on the backtracking algorithm. Using a single thread (which corresponds to a DFS in the search tree) leads to 140 of the 200 instances solved. As shown in Figure 9, using 10 or 100 threads improves the performance of the algorithm, for an additional 16 instances solved for 100 threads. However, increasing the number of threads to 1000 worsens the performance. This is because using more threads also increases the memory usage, which causes the algorithm to exceed the 10 GiB memory limit and to consequently abort on some instances. As expected, the number of threads makes no difference for no-instances, as threads do not shrink the search space. However, using more than one thread is often helpful for identifying yes-instances early.

6.5 Evaluation of the Best Configuration

Our experiments in the previous section indicate that the configuration using the branching strategy `KuratowskiMulti-1K` with filter `SC` and 100 threads performs the best.

Running this algorithm on all non-planar biconnected components of the `North` and `Rome` graphs with a time limit of 15 minutes shows a clear dependency between the running time and the size of the graph. While almost all instances up to 30 vertices are solved, this rate

Table 2 The results for the data sets `Bicomps-N` and `Bicomps-R`, consisting of the non-planar biconnected components of the `North` and `Rome` graphs, respectively, with a time limit of 15 minutes. Note that the medians for the number of processed nodes and the running time only take solved instances into account. The last two columns show the proportion of the total instances that were labeled as 1-planar and not 1-planar, respectively.

Group	#	Group Median			Solved		1-Planar %	
		Density	Proc. Nodes	Time (s)	#	%	Yes	No
`Bicomps-R` 1-10	9	1.57	2	0.0	9	100.0	100.0	0.0
`Bicomps-R` 11-20	250	1.47	3	0.0	250	100.0	100.0	0.0
`Bicomps-R` 21-30	1296	1.39	93	0.02	1295	99.9	99.8	0.1
`Bicomps-R` 31-40	1843	1.43	9404	4.57	1753	95.1	95.1	0.1
`Bicomps-R` 41-50	1358	1.44	23904	21.35	908	66.9	66.9	0.0
`Bicomps-R` 51-60	1227	1.44	43785	60.26	401	32.7	32.7	0.0
`Bicomps-R` 61-70	1126	1.45	46543	80.68	79	7.0	7.0	0.0
`Bicomps-R` 71-80	929	1.47	48832	116.08	16	1.7	1.7	0.0
`Bicomps-R` 81-90	214	1.45	44744	145.36	3	1.4	1.4	0.0
`Bicomps-R` 91-100	1	1.42	–	–	0	0.0	0.0	0.0
`Bicomps-R`	8253	1.44	7882	3.58	4714	57.1	57.1	0.0
`Bicomps-N` 1-10	56	2.05	5	0.0	56	100.0	92.9	7.1
`Bicomps-N` 11-20	124	2.0	52	0.01	124	100.0	60.5	39.5
`Bicomps-N` 21-30	94	1.9	607	0.26	93	98.9	48.9	50.0
`Bicomps-N` 31-40	52	1.81	192	0.25	41	78.8	50.0	28.8
`Bicomps-N` 41-50	30	1.48	1257	3.61	28	93.3	73.3	20.0
`Bicomps-N` 51-60	44	1.92	440	2.13	19	43.2	11.4	31.8
`Bicomps-N` 61-70	21	1.92	5216	9.5	2	9.5	4.8	4.8
`Bicomps-N` 71-80	2	1.58	5690	19.4	2	100.0	100.0	0.0
`Bicomps-N` 81-90	4	1.67	4088	12.29	2	50.0	50.0	0.0
`Bicomps-N` 91-100	2	1.52	47117	374.87	2	100.0	100.0	0.0
`Bicomps-N`	429	1.9	97	0.03	369	86.0	54.3	31.7

drops to around 60% for graphs of size 50. Overall, our algorithm solves 57% of all non-planar biconnected components of `Rome` graphs and 86% of those from the `North` graphs. Table 2 gives a detailed overview.

To compare our algorithm with the algorithm `1PlanarTester` of Binucci et al. [5], we also followed their experimental setup and ran our algorithm on all non-planar `North` and `Rome` graphs using the same time limit of three hours. Table 3 shows the details of the outcome. We note that, compared to the previous experiment, the correlation between running time and instance size is less pronounced as some instances may decompose into multiple smaller biconnected components that can be processed independently. Our implementation is able to solve all instances with up to 30 vertices within the time limit, with the median computing time being less than a second. While our algorithm still solves more than 95% of all instances with up to 50 vertices, this ratio notably decreases for larger graphs, especially for the `Rome` graphs. Here, we still solve around 60% of the instances with 61 to 70 vertices, but only 7% of the instances with 91 to 100 vertices. Overall, our implementation solves 61% of the 8523 non-planar `Rome` graphs. In contrast, we solve almost 90% of all non-planar `North` graphs – our algorithm even manages most of the instances with 71 to 100 vertices. The data in Table 3 shows that our algorithm significantly outperforms `1PlanarTester` on this dataset. For example, while `1PlanarTester` solves around 56% of all non-planar `North` graphs with up to 40 vertices, we solve around 99% of them. In fact, each instance of the data set that `1PlanarTester` solves within the three hour timelimit is solved in less than a second by our algorithm.

Table 3 Our results on the non-planar instances from the graph benchmark sets `North` and `Rome` with a time limit of three hours. Note that the medians for the number of processed nodes and the running time only take solved instances into account. For each group, the multicolumn "Solved" shows the number of instances we solved ("#"), the corresponding percentage of solved instances ("%"), and the percentage of instances the `1PlanarTester` by Binucci et al. [5] solved in the corresponding group within the same time limit ("% [5]"). For the latter, percentages in parantheses indicate that Binucci et al. [5] only evaluated a subset of the instances in the corresponding group. The last two columns show the proportion of the total instances that were labeled as 1-planar and not 1-planar, respectively.

Group	#	Group Median			Solved			1-Planar %	
		Density	Proc. Nodes	Time (s)	#	%	% [5]	Yes	No
`Rome` 10-20	91	1.5	6	0.0	91	100.0	91.2	100.0	0.0
`Rome` 21-30	164	1.36	14	0.0	164	100.0	69.5	100.0	0.0
`Rome` 31-40	1346	1.31	289	0.08	1342	99.7	(43.8)	99.2	0.5
`Rome` 41-50	1349	1.33	7730	3.59	1296	96.1	(37.8)	95.2	0.9
`Rome` 51-60	1054	1.33	17318	12.63	865	82.1	–	82.0	0.1
`Rome` 61-70	1102	1.34	28692	28.68	685	62.2	–	62.1	0.1
`Rome` 71-80	1023	1.33	47226	68.39	378	37.0	–	37.0	0.0
`Rome` 81-90	744	1.35	57530	90.5	125	16.8	–	16.8	0.0
`Rome` 91-100	1376	1.35	47334	82.41	97	7.0	–	7.0	0.0
`Rome` 101-110	4	1.28	–	–	0	0.0	–	0.0	0.0
`Rome`	8253	1.33	9279	4.58	5043	61.1	–	60.9	0.3
`North` 10-20	121	1.8	24	0.0	121	100.0	73.6	72.7	27.3
`North` 21-30	69	1.67	323	0.18	69	100.0	39.1	60.9	39.1
`North` 31-40	55	1.65	713	0.22	52	94.5	38.2	49.1	45.5
`North` 41-50	52	1.45	1058	0.64	51	98.1	(18.8)	65.4	32.7
`North` 51-60	48	1.89	2484	3.0	25	52.1	–	20.8	31.2
`North` 61-70	33	1.9	411	0.35	17	51.5	–	18.2	33.3
`North` 71-80	14	1.3	41	0.02	11	78.6	–	71.4	7.1
`North` 81-90	18	1.62	142	0.11	17	94.4	–	44.4	50.0
`North` 91-100	13	1.56	11557	14.15	10	76.9	–	38.5	38.5
`North`	423	1.73	142	0.03	373	88.2	–	54.4	33.8

7 Conclusion

We proposed a new backtracking framework for exact 1-planarity testing and investigated the impact of different branching strategies, filter criteria, and traversal strategies on its effectiveness. Our experiments show that some of our optimizations massively decrease the search space for such algorithms and thus greatly enhance their performance. In particular, using Kuratowski subdivisions to guide the search more than doubles the number of solved instances compared to the baseline that uses binary branching and our best filter, `SC`, rejects around 90% of the nodes in the search tree it is invoked on; it thus offers a significant, exponential decrease of the search space. Our best configuration combines this with threads that allow to search multiple parts of the search tree in parallel and altogether yields a practical 1-planarity testing and embedding algorithm that significantly outperforms its competitor `1PlanarTester` [5] and solves most instances from our test set with up to 50 vertices within seconds.

References

1 Patrizio Angelini, Michael A. Bekos, Michael Kaufmann, and Thomas Schneck. Efficient generation of different topological representations of graphs beyond-planarity. *Journal of Graph Algorithms and Applications*, 24(4):573–601, 2020. doi:10.7155/jgaa.00531.

2 Christopher Auer, Christian Bachmaier, Franz J. Brandenburg, Andreas Gleißner, Kathrin Hanauer, Daniel Neuwirth, and Josef Reislhuber. Outer 1-planar graphs. *Algorithmica*, 74(4):1293–1320, 2016. doi:10.1007/s00453-015-0002-1.

3 Christopher Auer, Franz J. Brandenburg, Andreas Gleißner, and Josef Reislhuber. 1-planarity of graphs with a rotation system. *Journal of Graph Algorithms and Applications*, 19(1):67–86, 2015. doi:10.7155/jgaa.00347.

4 Michael J. Bannister, Sergio Cabello, and David Eppstein. Parameterized complexity of 1-planarity. *Journal of Graph Algorithms and Applications*, 22(1):23–49, 2018. doi:10.7155/jgaa.00457.

5 Carla Binucci, Walter Didimo, and Fabrizio Montecchiani. 1-planarity testing and embedding: An experimental study. *Computational Geometry*, 108:101900, 2023. doi:10.1016/j.comgeo.2022.101900.

6 Thomas Bläsius, Marcel Radermacher, and Ignaz Rutter. How to draw a planarization. *J. Graph Algorithms Appl.*, 23(4):653–682, 2019. doi:10.7155/JGAA.00506.

7 Franz J. Brandenburg. Recognizing optimal 1-planar graphs in linear time. *Algorithmica*, 80(1):1–28, 2018. doi:10.1007/s00453-016-0226-8.

8 Christoph Buchheim, Dietmar Ebner, Michael Jünger, Gunnar W. Klau, Petra Mutzel, and René Weiskircher. Exact crossing minimization. In *13th International Symposium on Graph Drawing (GD' 05)*, volume 3843 of *Lecture Notes in Computer Science*, pages 37–48. Springer, 2005. doi:10.1007/11618058_4.

9 Sergio Cabello and Bojan Mohar. Adding one edge to planar graphs makes crossing number and 1-planarity hard. *SIAM Journal on Computing*, 42(5):1803–1829, 2013. doi:10.1137/120872310.

10 Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W. Klau, Karsten Klein, and Petra Mutzel. The open graph drawing framework (OGDF). In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization*, pages 543–569. Chapman and Hall/CRC, 2013.

11 Markus Chimani, Carsten Gutwenger, Petra Mutzel, and Christian Wolf. Inserting a vertex into a planar graph. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '09)*, pages 375–383. SIAM, 2009. doi:10.1137/1.9781611973068.42.

12 Markus Chimani, Max Ilsen, and Tilo Wiedera. Star-struck by fixed embeddings: Modern crossing number heuristics. In *29th International Symposium on Graph Drawing and Network Visualization (GD '21)*, volume 12868 of *Lecture Notes in Computer Science*, pages 41–56. Springer, 2021. doi:10.1007/978-3-030-92931-2_3.

13 Markus Chimani, Petra Mutzel, and Immanuel M. Bomze. A new approach to exact crossing minimization. In *Proceedings of the 16th Annual European Symposium on Algorithms (ESA '08)*, volume 5193 of *Lecture Notes in Computer Science*, pages 284–296. Springer, 2008. doi:10.1007/978-3-540-87744-8_24.

14 Markus Chimani, Petra Mutzel, and Jens M. Schmidt. Efficient extraction of multiple kuratowski subdivisions. In *Graph Drawing, 15th International Symposium, GD 2007*, volume 4875 of *Lecture Notes in Computer Science*, pages 159–170. Springer, 2007. doi:10.1007/978-3-540-77537-9_17.

15 Bruno Courcelle. The monadic second-order theory of graphs I: Recognizable sets of finite graphs. *Information and Computation*, 1990.

16 Éric Colin de Verdière and Petr Hlinený. FPT algorithms for crossing number problems: A unified approach. *CoRR*, abs/2410.00206, 2024. doi:10.48550/arXiv.2410.00206.

17 Éric Colin de Verdière and Thomas Magnard. An FPT algorithm for the embeddability of graphs into two-dimensional simplicial complexes. In *29th Annual European Symposium on*

Algorithms, ESA 2021, volume 204 of *LIPIcs*, pages 32:1–32:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ESA.2021.32.

18 Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Math. Program.*, 91(2):201–213, 2002. doi:10.1007/S101070100263.

19 Peter Eades, Seok-Hee Hong, Naoki Katoh, Giuseppe Liotta, Pascal Schweitzer, and Yusuke Suzuki. A linear time algorithm for testing maximal 1-planarity of graphs with a rotation system. *Theoretical Computer Science*, 513:65–76, 2013. doi:10.1016/j.tcs.2013.09.029.

20 Michael R Garey and David S Johnson. Crossing number is np-complete. *SIAM Journal on Algebraic Discrete Methods*, 4(3):312–316, 1983.

21 Alexander Grigoriev and Hans L Bodlaender. Algorithms for graphs embeddable with few crossings per edge. *Algorithmica*, 49(1):1–11, 2007. doi:10.1007/S00453-007-0010-X.

22 Carsten Gutwenger, Petra Mutzel, and René Weiskircher. Inserting an edge into a planar graph. *Algorithmica*, 41(4):289–308, 2005. doi:10.1007/S00453-004-1128-8.

23 Thekla Hamm and Petr Hlinený. Parameterised partially-predrawn crossing number. In *38th International Symposium on Computational Geometry (SoCG '22)*, volume 224 of *LIPIcs*, pages 46:1–46:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SoCG.2022.46.

24 Thekla Hamm, Fabian Klute, and Irene Parada. Computing crossing numbers with topological and geometric restrictions. *CoRR*, abs/2412.13092, 2024. doi:10.48550/arXiv.2412.13092.

25 Seok-Hee Hong, Peter Eades, Naoki Katoh, Giuseppe Liotta, Pascal Schweitzer, and Yusuke Suzuki. A linear-time algorithm for testing outer-1-planarity. *Algorithmica*, 72(4):1033–1054, 2015. doi:10.1007/s00453-014-9890-8.

26 Seok-Hee Hong and Takeshi Tokuyama, editors. *Beyond Planar Graphs*. Springer Singapore, 2020. doi:10.1007/978-981-15-6533-5.

27 Stephen G Kobourov, Giuseppe Liotta, and Fabrizio Montecchiani. An annotated bibliography on 1-planarity. *Computer Science Review*, 25:49–67, 2017. doi:10.1016/j.cosrev.2017.06.002.

28 Vladimir P Korzhik and Bojan Mohar. Minimal obstructions for 1-immersions and hardness of 1-planarity testing. *Journal of Graph Theory*, 72(1):30–71, 2013. doi:10.1002/JGT.21630.

29 Casimir Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamenta Mathematicae*, 15(1):271–283, 1930. URL: http://eudml.org/doc/212352.

30 Giordano Da Lozzo and Ignaz Rutter. Planarity of streamed graphs. *Theor. Comput. Sci.*, 799:1–21, 2019. doi:10.1016/J.TCS.2019.09.029.

31 Miriam Münch, Maximilian Pfister, and Ignaz Rutter. Exact and approximate k-planarity testing for maximal graphs of small pathwidth. In *Graph-Theoretic Concepts in Computer Science - 50th International Workshop, WG 2024*, volume 14760 of *Lecture Notes in Computer Science*, pages 430–443. Springer, 2024. doi:10.1007/978-3-031-75409-8_30.

32 Miriam Münch and Ignaz Rutter. Parameterized algorithms for beyond-planar crossing numbers. In *32nd International Symposium on Graph Drawing and Network Visualization, GD 2024*, volume 320 of *LIPIcs*, pages 25:1–25:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.GD.2024.25.

33 Petra Mutzel. The crossing number of graphs: Theory and computation. In *Efficient Algorithms, Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*, volume 5760 of *Lecture Notes in Computer Science*, pages 305–317. Springer, 2009. doi:10.1007/978-3-642-03456-5_21.

34 North and Rome graphs. URL: http://www.graphdrawing.org/data.html.

35 János Pach and Géza Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17(3):427–439, 1997. doi:10.1007/3-540-62495-3_59.

36 Sergey Pupyrev. OOPS: Optimized one-planarity solver via SAT. In *33nd International Symposium on Graph Drawing and Network Visualization, GD 2025*, volume 357 of *LIPIcs*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.GD.2025.14.

37 H. C. Purchase, R. F. Cohen, and M. I. James. An experimental study of the basis for graph drawing algorithms. *ACM Journal of Experimental Algorithmics*, 2:4, 1997. doi:10.1145/264216.264222.

38 Helen C. Purchase. Which aesthetic has the greatest effect on human understanding? In *Proceedings 5th International Symposium on Graph Drawing (GD '97)*, volume 1353 of *Lecture Notes in Computer Science*, pages 248–261. Springer, 1997. doi:10.1007/3-540-63938-1_67.

39 Helen C. Purchase, Jo-Anne Allder, and David A. Carrington. Graph layout aesthetics in UML diagrams: User preferences. *Journal of Graph Algorithms and Applications*, 6(3):255–279, 2002. doi:10.7155/jgaa.00054.

40 Marcel Radermacher, Klara Reichard, Ignaz Rutter, and Dorothea Wagner. Geometric heuristics for rectilinear crossing minimization. *ACM Journal of Experimental Algorithmics*, 24(1):1.12:1–1.12:21, 2019. doi:10.1145/3325861.

41 Marcel Radermacher and Ignaz Rutter. Geometric crossing-minimization - A scalable randomized approach. In *27th Annual European Symposium on Algorithms (ESA'19)*, volume 144 of *LIPIcs*, pages 76:1–76:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ESA.2019.76.

42 Marcel Radermacher and Ignaz Rutter. Inserting an edge into a geometric embedding. *Computational Geometry*, 102:101843, 2022. doi:10.1016/j.comgeo.2021.101843.

43 Marcus Schaefer. Picking planar edges; or, drawing a graph with a planar subgraph. In *Graph Drawing - 22nd International Symposium, GD 2014*, volume 8871 of *Lecture Notes in Computer Science*, pages 13–24. Springer, 2014. doi:10.1007/978-3-662-45803-7_2.

44 Marcus Schaefer. The graph crossing number and its variants: A survey. *The Electronic Journal of Combinatorics*, pages DS21–May, 2021. doi:10.37236/2713.

45 Yusuke Suzuki. 1-planar graphs. In *Beyond Planar Graphs: Communications of NII Shonan Meetings*, pages 47–68. Springer, 2020. doi:10.1007/978-981-15-6533-5_4.

Constrained Flips in Plane Spanning Trees

Oswin Aichholzer
Graz University of Technology, Austria

Joseph Dorfer
Graz University of Technology, Austria

Birgit Vogtenhuber
Graz University of Technology, Austria

Abstract

A flip in a plane spanning tree T is the operation of removing one edge from T and adding another edge such that the resulting structure is again a plane spanning tree. For trees on a set of points in convex position we study two classic types of constrained flips: (1) Compatible flips are flips in which the removed and inserted edge do not cross each other. We relevantly improve the previous upper bound of $2n - O(\sqrt{n})$ on the diameter of the compatible flip graph to $\frac{5n}{3} - O(1)$, by this matching the upper bound for unrestricted flips by Bjerkevik, Kleist, Ueckerdt, and Vogtenhuber [SODA 2025] up to an additive constant of 1. We further show that no shortest compatible flip sequence removes an edge that is already in its target position. Using this so-called happy edge property, we derive a fixed-parameter tractable algorithm to compute the shortest compatible flip sequence between two given trees. (2) Rotations are flips in which the removed and inserted edge share a common vertex. Besides showing that the happy edge property does not hold for rotations, we improve the previous upper bound of $2n - O(1)$ for the diameter of the rotation graph to $\frac{7n}{4} - O(1)$.

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics; Mathematics of computing → Combinatorics; Mathematics of computing → Combinatoric problems

Keywords and phrases Non-crossing spanning trees, Flip Graphs, Diameter, Complexity, Happy edges

Digital Object Identifier 10.4230/LIPIcs.GD.2025.5

Related Version *arXiv Version*: `https://arxiv.org/abs/2508.15520` [2]

Funding *Joseph Dorfer*: Austrian Science Foundation (FWF) 10.55776/DOC183.

1 Introduction

Let S be a finite point set in the plane in general position, that is, no three points lie on a common line. We call S a *convex* point set if no point in S lies in the interior of the convex hull of S. A plane straight-line graph of S is a graph with vertex set S and whose edges are straight line segments between pairs of points of S such that no two edges intersect, except at a common endpoint. All graphs considered in this paper are straight-line graphs. For brevity we will omit the term straight-line. A *plane spanning tree* is a plane connected graph with $n - 1$ edges. In a convex point set, edges appear in two forms. *Convex hull edges* are edges that lie on the boundary of the convex hull of the point set. Edges that are not convex hull edges are *diagonals*.

Flip Graphs of Plane Spanning Trees. A *flip* between two plane spanning trees of S is the operation that removes an edge from a tree and adds another edge such that the resulting structure is again a plane spanning tree. We also denote this operation as an *unrestricted flip*. A constrained version of flips are so-called *compatible flips*, for which the removed edge and the added edge are not allowed to cross. *Rotations* are a restricted subclass of compatible flips for which we additionally require the removed edge and the added edge to share a common


© Oswin Aichholzer, Joseph Dorfer, and Birgit Vogtenhuber;
licensed under Creative Commons License CC-BY 4.0



vertex. The most restricted and local operation is a so called *slide*. A slide is an operation in which the added edge (u, v) and the removed edge (v, w) are again required to share a common vertex, additionally, the triangle $\Delta(u, v, w)$ must not contain any other vertex and the edge (u, w) has to be contained in the current tree. This can visually be interpreted as sliding the edge (u, v) along the edge (u, w) into the edge (v, w). The *(compatible) flip graph* of plane spanning trees of S has as its vertex set all such trees. Two vertices T_1, T_2 of this flip graph are connected with an edge if and only if T_1 can be transformed into T_2 by a single (compatible) flip. See Figure 1 for an example of flips in plane spanning trees.

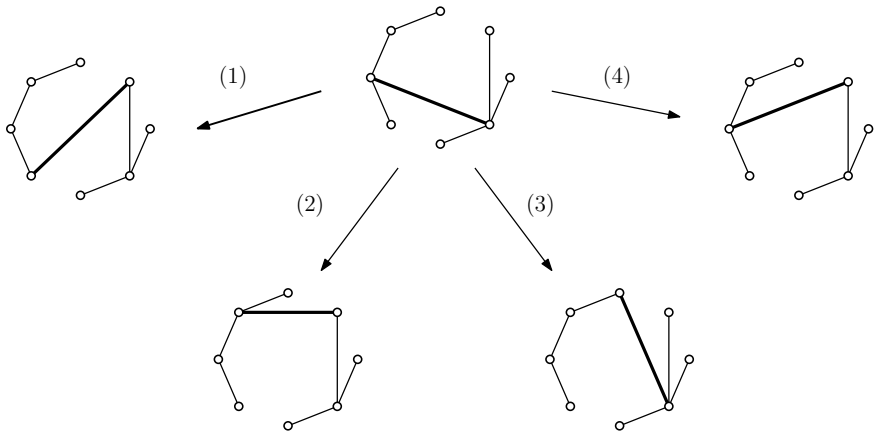


Figure 1 Four flips in a plane spanning tree. (1) unrestricted flip, (2) compatible flip, (3) rotation, (4) slide.

Given an initial tree T_{in} and a target tree T_{tar}, a *(compatible) flip sequence* from T_{in} to T_{tar} is a path from T_{in} to T_{tar} in the (compatible) flip graph. The *(compatible) flip distance* between T_{in} and T_{tar} is the length of a shortest path between T_{in} and T_{tar} in the flip graph. Any (compatible) flip sequence of this length is called a *shortest (compatible) flip sequence*. Similarly, we define the *rotation graph*, *rotation sequences* and the *rotation distance*. See Figure 2 for an illustration of the concepts.

For graph reconfiguration problems three natural questions arise: (1) Is the flip graph connected? (2) What is the diameter of the flip graph? (3) What is the complexity of computing shortest flip sequences between two specific configurations?

Connectedness and Diameter. For any n-point set S, the flip graph of plane spanning trees on S is known to be connected and has radius exactly $n-2$. The same holds for the compatible flip graph and the rotation graph; see [5, 6]. Hence, the diameter always lies between $n-2$ and $2n-4$. Since a lower bound for the diameter of $\lfloor \frac{3}{2}n \rfloor - 5$ for convex n-point sets was shown in [12], the flip graph of plane spanning trees on convex point sets has received considerable attention. However, despite a prevailing conjecture that the diameter is at most $\frac{3}{2}n + O(1)$ [10, 14], even improving the upper bound to $2n - o(n)$ took surprisingly long. In the last few years, the upper bound for its diameter was improved to $2n - \Omega(log(n))$ in [1] and soon after to $2n - \Omega(\sqrt{n})$ in [10]. The upper bound from [10] is constructive and, though not stated explicitly, only uses compatible flips. Hence it also provides an upper bound for the diameter of the according compatible flip graph. In [8], the diameter was bounded from above by cn with $c = \frac{1}{12}\left(22 + \sqrt{2}\right) \approx 1.95$, which marked the first linear improvement over the initial bound from [5]. Very recently, a lower bound of $\frac{14}{9}n - O(1)$ and an upper bound of $\frac{5}{3}n - 3$ on the diameter were achieved in [6], where the upper bound in general requires

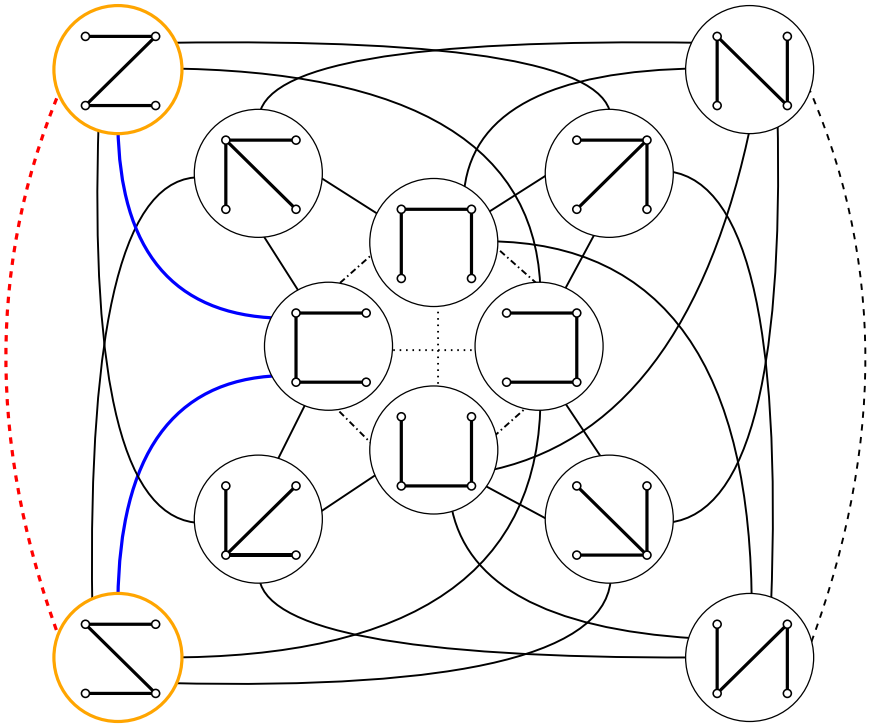


Figure 2 Flip graph of plane spanning trees on four vertices in convex position. Dashed edges denote flips that are part of the flip graph, but not the compatible flip graph. The dotted lines are compatible flips, but not rotations and the dash dotted lines are rotations but not slides. One pair of two plane spanning trees is marked in orange. Their shortest flip sequence of length one is marked in red (thicker dashed). A shortest compatible flip sequence of length two is marked in blue (thicker).

non-compatible flips. It is also known that any plane spanning tree can be transformed into any other using $\Theta(n^2)$ many slides [4]. For a survey on further flip operations in trees with multiple flips in parallel as well as flips with labeled edges, we refer to [18].

Happy Edges and Complexity. For any graph reconfiguration problem, *happy edges* are edges that lie in both the initial and the target graph. A flip graph fulfills the *happy edge property* if there exists a shortest flip sequence between any two graphs that never flips happy edges. The happy edge property often is a good indication for the complexity of a graph reconfiguration problem. For example, for triangulations of simple polygons [3] and general point sets [16, 19], finding shortest flip sequences is computationally hard. The hardness proofs use counterexamples to the happy edge property as a key ingredient. Conversely, the happy edge property is known to hold for triangulations of convex polygons [20]. Though the complexity of finding shortest flip sequences is still open, the happy edge property yields multiple fixed-parameter tractable algorithms [7, 11, 15, 17]. Moreover, the happy edge property holds for plane perfect matchings of convex point sets and shortest flip sequences can be found in polynomial time [13]. For plane spanning trees in convex position the happy edge property was conjectured to hold [1, 14] for different types of flips. Up to now, it has only been disproved for slides [1].

Contribution and Outline. We will start by improving upper bounds on the flip graph for restricted flip types in convex point sets. In Section 2 we improve the upper bound on the diameter of the compatible flip graph from $2n - \sqrt{n}$ in [10] to $\frac{5}{3}(n-1)$. Section 3 is dedicated to improving the upper bound on the rotation graph from $2n - O(1)$ in [5] to $\frac{7}{4}(n-1)$.

Afterwards we deal with the happy edge property. In Section 4 we provide a more refined framework for happy edge properties and in Section 5 we prove the happy edge property for compatible flips in convex point sets and provide a counterexample to the happy edge property for rotations.

Finally, in Section 6 we use variants of the happy edge property to show fixed-parameter tractability of the flip distance for both unrestricted and compatible flips.

Due to space restrictions, full technical details and proofs for statements marked with ($\star$) *are given in the arXiv Version of our paper [2].*

2 Bounding the Diameter of the Compatible Flip Graph

This section is devoted to improving the upper bound on the diameter of the compatible flip graph. All the currently known improvements to the upper bound of the length of shortest flip sequences for unrestricted flips are based on two different ways to handle edges: They either invest two flips to flip an edge from $T_{\text{in}} \setminus T_{\text{tar}}$ to a convex hull edge and later flip that convex hull edge to an edge of $T_{\text{tar}} \setminus T_{\text{in}}$, or perform a single *perfect flip*, that is, a flip from an edge of $T_{\text{in}} \setminus T_{\text{tar}}$ to an edge of $T_{\text{tar}} \setminus T_{\text{in}}$. Upper bounding the length of a flip sequence is then achieved by lower bounding the number of perfect flips. Note that any such flip sequence induces a pairing of the edges of $T_{\text{in}} \setminus T_{\text{tar}}$ with edges of $T_{\text{tar}} \setminus T_{\text{in}}$.

To obtain our improved bound for the diameter of the compatible flip graph, we build on strategies for bounding the diameter of the flip graph with unrestricted flips from [6]. We give an overview of their proof and highlight the parts where we provide adjustments to make the flip sequence compatible.

The pairing of edges in [6] is obtained as follows: The authors consider the linear representation of a tree T on a convex point set as illustrated in Figure 3. The linear representation is obtained by cutting the convex point set between two consecutive vertices on the convex hull and unfolding the tree T such that its vertices are on a straight line.

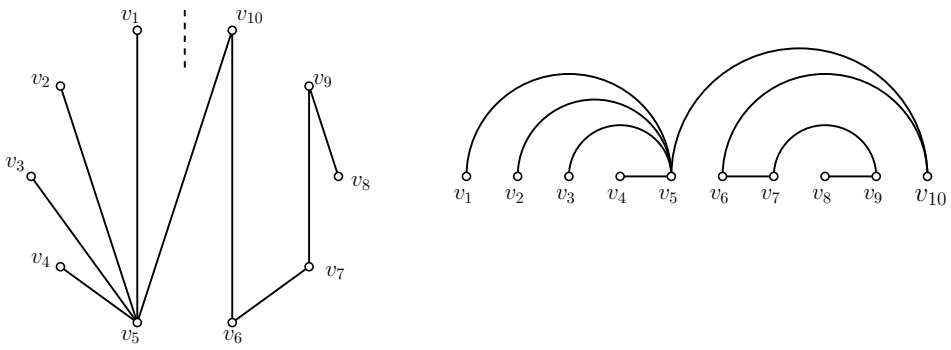


Figure 3 A tree T on a convex point set (left) and a linear representation of T (right).

The linear representation induces a linear order $v_1, \ldots, v_n$ of the vertices of T. With this linear order, the *length of an edge* $e = (v_i, v_j)$, where $i < j$, is defined as $j - i$. Further, the edge e *covers* all vertices $\{v_i, \ldots, v_j\}$ as well as any edge (v_k, v_ℓ) with $k, \ell \in \{i, \ldots, j\}$.

A *gap* $\{v_i, v_{i+1}\}$ is an edge between two consecutive vertices in the linear order (which might or might not be present in the tree T). Next, the authors define a bijection (Lemma 3.1 in [6]) between *gaps* and edges of T, where each gap is in bijection with the shortest edge of T that covers that gap. The edges are then partitioned into categories based on how many vertices they share with their corresponding gap (v_i, v_{i+1}); see Figure 4 for an illustration.

Concretely, an edge (u, v) is a ...

- *short edge* if $\{u, v\} = \{v_i, v_{i+1}\}$
- *near edge* if $|\{u, v\} \cap \{v_i, v_{i+1}\}| = 1$
- *wide edge* if $|\{u, v\} \cap \{v_i, v_{i+1}\}| = 0$

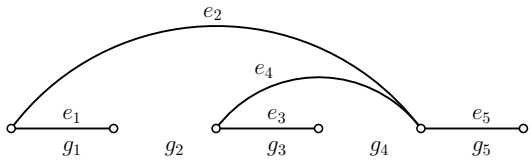


Figure 4 Edges e_1, e_3 and e_5 are short edges for gaps g_1, g_3 and g_5 respectively, e_4 is a near edge for g_4, and e_2 is a wide edge for g_2.

The authors show that the number of short edges in any tree T is always larger than the number of wide edges. In a next step, they construct an edge-edge-bijection between the edges in T_{in} and T_{tar} by concatenating the edge-gap-bijection for T_{in} with the gap-edge-bijection of T_{tar}. An edge $e \in T_{\text{in}}$ and its image from the edge-edge-bijection $e' \in T_{\text{tar}}$ form a *pair*. If an edge e is in both T_{in} and T_{tar}, and e is in bijection to itself, then e is not flipped at all. Note that any edge can be flipped to its assigned gap by a single (compatible) flip. The authors use this and the relation between the numbers of short and wide edges in each tree to show that all pairs of edges containing at least one short or wide edge can be flipped using an average of $\frac{3}{2}$ flips per pair. This is done by flipping every edge in such a pair to its corresponding gap and later from its gap to its target edge, which takes two flips minus the number of short edges in the pair.

The remaining pairs then only consist of near edges. These pairs can further be partitioned into three sets as illustrated in Figure 5.

- *above pairs*, where e and e' share a vertex and e is longer than e'.
- *below pairs*, where e and e' share a vertex and e is shorter than e'.
- *crossing pairs*, where e and e' cross, or equivalently, e and e' share a different vertex with their gap.

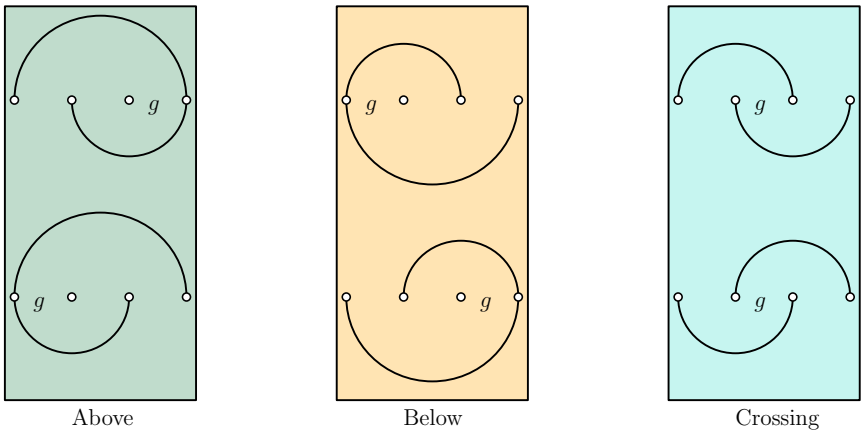


Figure 5 Illustration of above, below and crossing pairs of edges. Edges of T_{in} are drawn above the vertices, while edges of T_{tar} are drawn below the vertices. The respective gap is denoted by g.

The sets of above, below and crossing pairs are then denoted by A, B, and C, respectively. The authors provide a flip sequence that flips all pairs in A (resp. B, C) directly [6, Lemma 3.2]. Since a direct flip of an above or below pair is compatible, this implies the following lemma.

▶ **Lemma 1.** *Assume that in the linear representation, the trees T_{in} and T_{tar} consist only of short edges and pairs of edges in A (resp. B). Then there is a compatible flip sequence from T_{in} to T_{tar} of length $|A|$ (resp. $|B|$).*

In contrast, no direct flip between two edges of a pair in C is compatible, since any such pair is crossing. In the arXiv version of our paper [2] we develop an approach to flip these edges such that the resulting flip sequence is compatible while using only one additional flip.

▶ **Lemma 2 ($\star$).** *Assume that in the linear representation, the trees T_{in} and T_{tar} consist only of short edges and pairs of edges in C. Then there is a compatible flip sequence from T_{in} to T_{tar} of length $|C| + 1$.*

To obtain their upper bound on the flip distance between two trees T_{in}, T_{tar}, the authors cut the original instances T_{in}, T_{tar} along happy diagonals, handle each of the resulting subinstances separately, and then add up the flips needed in each of the subinstances. Let b denote the number of happy convex hull edges, c the number of happy diagonals, and d the cardinality of $T_{\text{in}} \setminus T_{\text{tar}}$. Using that $b + c + d = n - 1$, their resulting bound [6, Theorem 4.1] is $\frac{5}{3}d + \frac{2}{3}b - \frac{4}{3} \leq \frac{5}{3}(n-1) - \frac{4}{3}$. With similar calculations, we show the following theorem for compatible flips in the arXiv version of our paper [2].

▶ **Theorem 3 ($\star$).** *Let T_{in} and T_{tar} be two plane spanning trees in convex point sets with n vertices. Then, there exists a compatible flip sequence from T_{in} to T_{tar} of length at most $\frac{5}{3}d + \frac{2}{3}b + c - \frac{1}{3} \leq \frac{5}{3}(n-1) - \frac{1}{3}$, where $d = |T_{in} \setminus T_{tar}| = \frac{|T_{in} \Delta T_{tar}|}{2}$ and where b and c denote the number of happy convex hull edges and happy non-convex-hull edges, respectively.*

We remark that our bound with respect to the parameters d, b, and c can be relevantly higher than the one in [6, Theorem 4.1] if the two trees share many happy diagonals. This behaviour was not unexpected due to the following result: In [9, Theorem 1.6], the authors provide examples of pairs of plane spanning trees with (unrestricted) flip distance d and compatible flip distance $2d$. The according tree pairs have $c \approx \frac{n}{2}$ happy diagonals. However, any pair of trees that could reach the upper bounds in [6, Theorem 4.1] and Theorem 3 must have $c = 0$. So while for certain instances the flip distance and the compatible flip distance may differ strongly, extremal examples that determine the diameter are not affected.

3 Bounding the Diameter of the Rotation Graph

Next, we show that for the even more constrained rotation as flip operation, the diameter of the resulting flip graph is linear with leading coefficient less than two.

For this purpose we again construct a pairing of edges such that a perfect flip can be performed for many of them. The challenge that arises, compared to unrestricted or compatible flips, is that rotations require the removed and the added edge to share a vertex. This rules out the gap based pairing from the last section, since in crossing pairs, edges do not share vertices and the same holds for most pairs that contain a wide edge.

▶ **Lemma 4.** *For a plane spanning tree T on a point set S and a vertex $v \in S$ there exists a bijection $\rho_{T,v}$ between vertices in $S \setminus \{v\}$ and edges in T such that $\rho_{T,v}(w)$ is incident to w for every vertex $w \in S \setminus \{v\}$.*

Proof. We root the tree T at v and orient every edge away from v such that v forms the unique source of the oriented graph; see Figure 6 for an illustration. Now map each vertex in $S \setminus \{v\}$ to its incoming edge. ◀

▶ **Corollary 5.** *Let T_1 and T_2 be two plane spanning trees on a point set S and let v be a vertex in S. Then there exists a pairing of edges from T_1 with edges in T_2 such that:*

- *Two edges that are paired together share a vertex.*
- *Different pairs of edges share a different vertex.*
- *No pair of edges has v as its shared vertex.*

Proof. The pairing $\{(\rho_{T_1,v}(w), \rho_{T_2,v}(w) \mid w \in S \setminus \{v\}\}$ induced by the bijection in Lemma 4 has all the desired properties. ◀

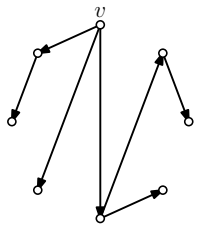


Figure 6 Illustration of the bijection from Lemma 4. All edges are oriented away from the root v. Each vertex in $S \setminus \{v\}$ is in bijection with its unique incoming edge.

For any two given plane spanning trees T_{in} and T_{tar}, we apply the bijection from Corollary 5 with $v := v_n$ to obtain a pairing of the edges of the two trees with all the mentioned properties. For an edge e that is in bijection with a vertex v_i we say that e is *attached* to v_i. Note that each vertex v_i with $i \in \{1, \ldots, n-1\}$ has an attached edge from each of T_{in} and T_{tar}, while v_n has no attached vertices.

From this point on we consider the two trees presented in their linear representation, see Figure 3, such that v_n is the rightmost vertex. The pairs of edges $(e_{\text{in}} = (v_i, v_j), e_{\text{tar}} = (v_i, v_k))$, where v_i is the vertex where edges e_{in} and e_{tar} are attached to, can now be subdivided into multiple classes. We say that a pair of edges is ...

- *right-attached above* if $j \leq k < i$
- *left-attached above* if $i < k \leq j$
- *right-attached below* if $k \leq j < i$
- *left-attached below* $i < j \leq k$
- *diving* if $j < i < k$
- *jumping* if $k < i < j$

We denote the respective sets of pairs in that order by RA, LA, RB LB, D and J. Additionally, we set $L = LA \cup LB$ and $R = RA \cup RB$. See Figure 7 for an illustration of the six types of edge pairs, as well as the two groups L and R. We remark that happy edges that are left-attached fulfill both, the definition for left-attached above and the one left-attached below. This does not pose any problem, since in the parts of the proof where this is relevant, we will only consider the whole set L. By symmetry, the same can be said about right-attached happy edges. Another noteworthy observation is that we do not handle convex hull edges any differently than diagonals. There might even be happy convex hull edges that belong to different pairs. This will lead to cases in which we rotate happy (convex-hull) edges. We allow this on purpose, since we will see in Section 5 that the happy edge property does not hold for rotations.

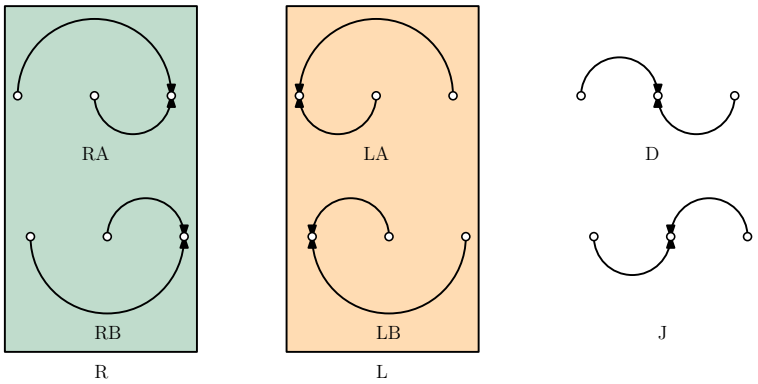


Figure 7 Illustration of pairs of edges in the sets RA, RB, R, LA, LB, L, D and J. Edges of T_{in} are drawn above the vertices, while edges of T_{tar} are drawn below the vertices. Each edge points to the vertex which it is attached to.

The new upper bound on the diameter of the rotation graph is described by the following Theorem.

▶ **Theorem 6.** *There exists a rotation sequence from T_{in} to T_{tar} of length at most* $2(n-1) - max\{|L|, |R|, |J|, |D|\} \leq \frac{7}{4}(n-1)$

We will use Lemma 7 below to get the edges from three of the sets L, R, J, and D "out of the way". Then we will show that all edges from the remaining set can be perfectly flipped.

We pair the edges and vertices of T via the bijection ρ_{T,v_n}. Further, for an arbitrary $j \in \{1, \ldots, n-1\}$, we assign a gap to every vertex except v_n by the following function (indices taken modulo n).

$$\delta_j : v_k \mapsto \begin{cases} (v_k, v_{k+1}) & \text{if } k > j \\ (v_{k-1}, v_k) & \text{if } k \leq j \end{cases}$$

In other words, we choose one particular gap (v_j, v_{j+1}) that is not assigned to any vertex and the other gaps are assigned such that they contain their assigned vertex.

In the following lemma, j is an arbitrary index, I is a subset of the convex hull edges in T that behave "nicely" with respect to δ_j, and K are all edges of T that are incident to v_n. I^* and K^* are according subsets that are required to be present in the resulting tree T^*.

▶ **Lemma 7.** *For any $j \in \{1, \ldots, n-1\}$, let $I = \{k \in \{1, \ldots, n-1\} \mid \rho_{T,v_n}(v_k) = \delta_j(v_k)\}$ and $K = \{k \in \{1, \ldots, n-1\} \mid \rho_{T,v_n}(v_k) = (v_n, v_k)\}$. Further, let I^*, K^* be a partition of $\{1, ..., n-1\}$, that is, $I^* \cap K^* = \emptyset$ and $I^* \cup K^* = \{1, \ldots, n-1\}$. Then there exists a rotation sequence from T to T^* of length at most $n - 1 - |I \cap I^*| - |K \cap K^*|$ that does not flip edges in $\rho_{T,v_n}(\{v_k \mid k \in (I \cap I^*) \cup (K \cap K^*)\})$ and T^* has the following properties:*

- *For all $k \in I^*$, the tree T^* contains the convex hull edges $\delta_j(v_k)$. That is, the convex hull edges attached to v_k directed away from v_n with respect to the order of vertices along the path on the convex hull from v_n to (v_j, v_{j+1})[1].*
- *The tree T^* contains the edges (v_n, v_k) for all $k \in K^*$. That is, all vertices that are attached to v_k are connected directly to v_n via edges that form a fan at v_n.*

[1] We remark that this direction in general differs from the orientation induced by the tree that is used in the proof of Lemma 4.

We refer to Figure 8 for an illustration of Lemma 7 applied to the tree from Figure 6.

Proof. We set $E_0 = \rho_{T,v_n}(\{v_k \mid k \in (I \cap I^*) \cup (K \cap K^*)\})$ and $T_0 = T \setminus E_0$. Now consider the forest $T_{0,vis}$ of edges that are *visible* from v_n in T_0. These are all edges for which no straight-line segment from v_n to the edge is intersected by any other edge. Further, let v_k be a vertex of degree one in $T_{0,vis}$. Then the only edge incident to v_k in $T_{0,vis}$ is $\rho_{T,v_n}(v_k)$. In particular, if $k \in I^*$ then $\delta_j(v_k)$ does not contain any edge and is visible from v_n, and if $k \in K^*$ then the straight-line segment between v_n and v_k is uncrossed.

We can execute a flip that exchanges the edge $\rho_{T,v_n}(v_k)$ with the edge $\delta_j(v_k)$ (if $k \in I^*$) or (v_n, v_k) if $k \in K^*$ to get a valid tree. We replace $E_0 \leftarrow E_0 \cup \{\delta_j(v_k)\}$ (resp. $E_0 \leftarrow E_0 \cup \{(v_n, v_k)\}$) and $T_0 \leftarrow T_0 \setminus \{\delta_j(v_k)\}$ (resp. $T_0 \leftarrow T_0 \setminus \{(v_n, v_k)\}$) and repeat the process. We repeat until $E_0 = T$. This takes at most $n - 1 - |I \cap I^*| - |K \cap K^*|$ rotations, since the cardinality of E_0 increases by one in every iteration. ◀

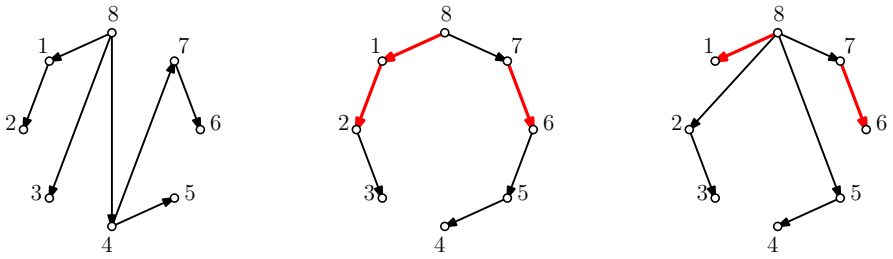


Figure 8 Illustration of Lemma 7. Left: The tree from Figure 6 with $n = 8$. Middle: Target structure of Lemma 7 with $K^* = \emptyset$ and chosen index $j = 3$. The edges in $\rho_{T,v_n}(I \cap I^*) = \{(v_8, v_1), (v_1, v_2), (v_7, v_6)\}$ that do not get rotated when rotating from the left to the middle tree are marked in red. Right: Target structure of Lemma 7 with $K^* = \{2, 5\}$ and the same index $j = 3$ as before. The edges in $\rho_{T,v_n}(I \cap I^*) = \{(v_8, v_1), (v_7, v_6)\}$ that do not get rotated when rotating from the left to the right tree are marked in red.

A gap (v_i, v_{i+1}) in a tree T is called *empty* if the edge (v_i, v_{i+1}) is not part of T. Note that in the target tree T^* from Lemma 7, the gap (v_j, v_{j+1}) is empty. Moreover, by setting $K^* = \emptyset$, a flip sequence as in Lemma 7 can be used to rotate all edges to the convex hull such that all convex hull edges that point away from v_n do not have to be rotated.

3.1 The Sets L and R

▶ **Proposition 8.** *There exists a flip sequence from T_{in} to T_{tar} that uses $2(n-1) - max\{|L|, |R|\}$ rotations.*

Without loss of generality, let $|R| > |L|$. For the sets R, we will (1) rotate all edges except the ones in R to the convex hull such that every edge in R has an empty gap to the left of its attachment point, (2) rotate all edges in pairs in R to their target position, and (3) obtain the final tree. Steps (1) and (3) are described in Lemma 9, Step (2) happens in Lemma 10. An illustration of the strategy can be seen in Figure 9.

▶ **Lemma 9** (⋆). *There exists a rotation sequence of length at most $n - 1 - |R|$ that rotates all edges except the ones in R to the convex hull such that each edge is rotated into the gap in the convex hull that is left of its assigned vertex. In particular, for an edge $e = (v_i, v_r)$ with $i < r$ that belongs to a pair in R, the gap (v_{r-1}, v_r) in the convex hull is empty. A symmetric argument holds for pairs in L.*

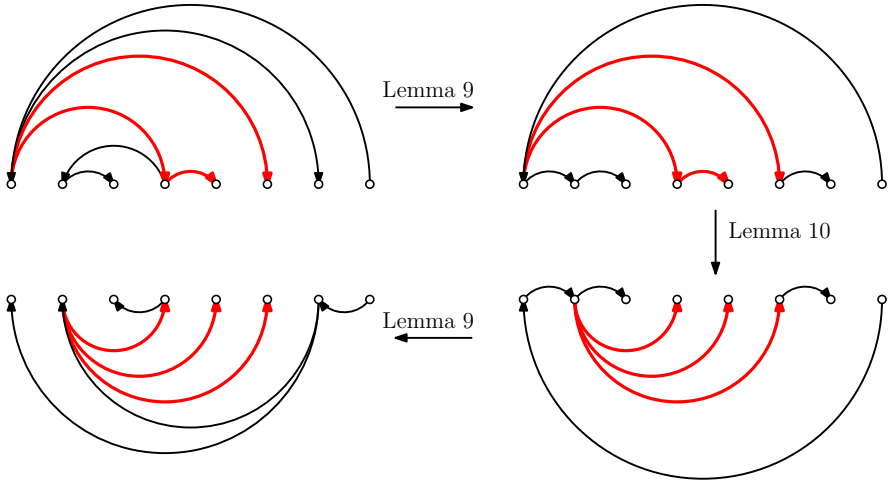


Figure 9 Steps of the proof of Proposition 8. First, rotate all edges except the ones in R (marked in red) to the convex hull, then rotate the pairs in R. At last, rotate all remaining edges from the convex hull back in.

Lemma 9 is an iterative application of Lemma 7. We apply the lemma with $K^* = \emptyset$ to every subtree that results from cutting the tree along edges in R (resp. L). Let T_1 and T_2 be the trees obtained from applying Lemma 9 to T_{in} and T_{tar}.

▶ **Lemma 10** ($\star$). *There exists a rotation sequence from T_1 to T_2 of length at most $|R|$ (resp. $|L|$).*

The proof of Lemma 10 reduces to the notion of conflict graphs from [6]. In Section 2, left-attached and right attached pairs have been combined to above and below pairs, but we also get an acyclic conflict graph, if we pair them according to whether they are right-attached of left-attached.

3.2 The Sets D and J

▶ **Proposition 11.** *There exists a flip sequence from T_{in} to T_{tar} that uses $2(n-1) - max\{|D|, |J|\}$ rotations.*

As the first rotation of the rotation sequence we rotate $\rho_{T_{\text{in}},v_n}(v_1)$ into the edge (v_1, v_n). From now on, we say that a *gap is visible from above* if the gap is empty and shares a *face* in the tree (more specifically a face of the drawing of T_{in} combined with the convex hull) with (v_1, v_n). The concept is illustrated in Figure 10. Note that there is always a unique gap that is visible from above and that it has an edge e_ℓ that is attached to its left vertex and (maybe) an edge e_r that is attached to its right vertex. e_r may not exist if g is the rightmost gap. Further, we show that e_ℓ and e_r share a face with (v_1, v_n).

▶ **Lemma 12.** *e_ℓ and e_r share a face with (v_1, v_n).*

Proof. Assume v_1 and v_n share a face with k additional vertices. Then it also has to share a face with k further edges in order for T to be a tree. Those k edges are exactly the edges that are attached to the k vertices and, thus, include e_ℓ and e_r. ◀

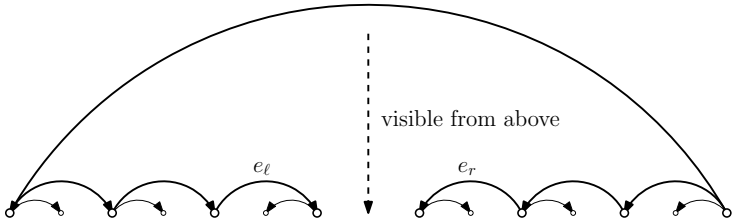


Figure 10 Gap that is visible from above.

▶ **Lemma 13** (⋆). *There exists a flip sequence of length at most $n-1$ that transforms T_{in} into a tree T^* that only contains the following three groups of edges:*

(A) *some convex hull edges attached to their right vertex,*

(B) *all target edges of jump pairs, and*

(C) *some edges that are attached to their left vertex and have as a second vertex a vertex of a jump pair or the vertex v_n.*

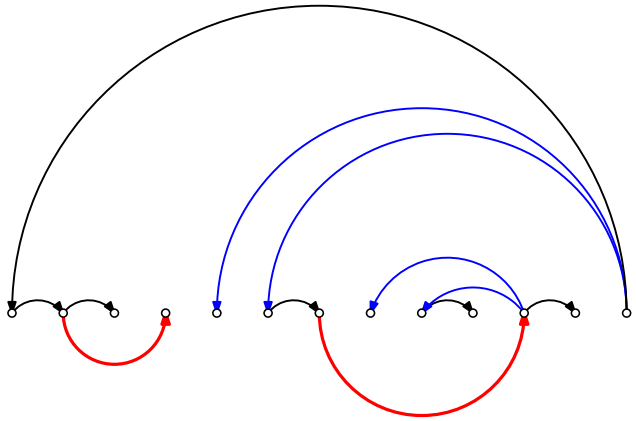


Figure 11 An intermediate tree T^* as produced by Lemma 13. The target edges from jump pairs are marked in red. The edges of type C, that from the set K^* in Lemma 14 are marked in blue.

A tree T^* as obtained from Lemma 13 is depicted in Figure 11. The edges that are of special interest for the further application of Lemma 14 are highlighted with colors.

▶ **Lemma 14** (⋆). *We can transform T_{tar} into T^* as in Lemma 13 in at most $n-1-|J|$ rotations.*

To obtain Lemma 14, we apply Lemma 7 to every face that results from cutting the tree along edges in pairs in J.

Combining the rotation sequence from Lemma 13 (going from T_{in} to T^*) and the reverse rotation sequence from Lemma 14 (going from T^* to T_{tar}), we obtain a rotation sequence between T_{in} and T_{tar} of length at most $2(n-1)-|J|$, which proves Proposition 11.

4 Relations of Happy Edge Properties

In this section, we introduce a more refined distinction of happy edge properties.

▶ **Definition 15.** *A graph reconfiguration problem, in which flips exchange edges, fulfills the*
- (weak) happy edge property*, if, from any initial graph G_{in} to any target graph G_{tar}, there exists a shortest flip sequence that does not flip happy edges.*
- strong happy edge property*, if, from any initial graph G_{in} to any target graph G_{tar}, any shortest flip sequence does not flip happy edges.*
- perfect flip property *if, whenever we can perform a flip f in G_{in} such that the resulting graph G_1 has one edge more in common with G_{tar} than G_{in} has in common with G_{tar}, then there exists a shortest flip sequence from G_{in} to G_{tar} that has G_1 as its first intermediate graph (that is, the flip sequence starts with f and f is called a* perfect flip*).*

Further, we provide a framework, that allows us to compare the different properties.

▶ **Proposition 16** (⋆)**.** *Let P be a graph reconfiguration problem where flips exchange one edge at a time.*

(i) *If any flip sequence in P that flips at least one happy edge can be shortened by at least two flips, then P fulfills the perfect flip property.*

(ii) *If for any flip in P from G to $(G \setminus \{e_1\}) \cup \{e_2\}$, e_1 and e_2 can never be in the same configuration, then the reverse direction in (i) holds.*

We remark that all the introduced properties may or may not hold for certain graph reconfiguration problems. While the condition in Proposition 16(ii) is not fulfilled for trees, it does hold for example for triangulations.

5 Happy Edges in Plane Spanning Trees on Convex Sets

In [1], the authors formulated the *Weak Happy Edge Conjecture* for trees on convex point sets:

▶ **Conjecture 17** (Conjecture 17 in [1])**.** *For any two plane spanning trees T_{in} and T_{tar} on a convex point set, there is a shortest flip sequence from T_{in} to T_{tar} that does not flip happy edges.*

Based on an example from [1] for a different context, we first observe that the perfect flip property holds neither for unrestricted flips nor for compatible flips on trees.

In contrast, we will show in this section that the strong happy edge property does hold for compatible flips on trees. Our first proof ingredient is the following lemma, which is an extension of [1, Proposition 18] for compatible flips.

▶ **Lemma 18** (⋆)**.** *Consider any point set S and any two plane spanning trees T_{in} and T_{tar} on S and any shortest compatible flip sequence from T_{in} to T_{tar}. If some edge e is removed and later added back, then some flip during that subsequence must add an edge f that crosses e.*

In [1], a *parking edge* is defined as an edge that appears in a flip sequence and that is not contained in $T_{\text{in}} \cup T_{\text{tar}}$. A second ingredient of our proof is Lemma 20, which verifies the compatible flip analogue of the following conjecture from [1].

▶ **Conjecture 19** (Parking Edge Conjecture, Conjecture 21 in [1])**.** *For any convex point set S and any two plane spanning trees T_{in} and T_{tar} on S, there is a shortest flip sequence from T_{in} to T_{tar} that only uses parking edges from the boundary of the convex hull of S.*

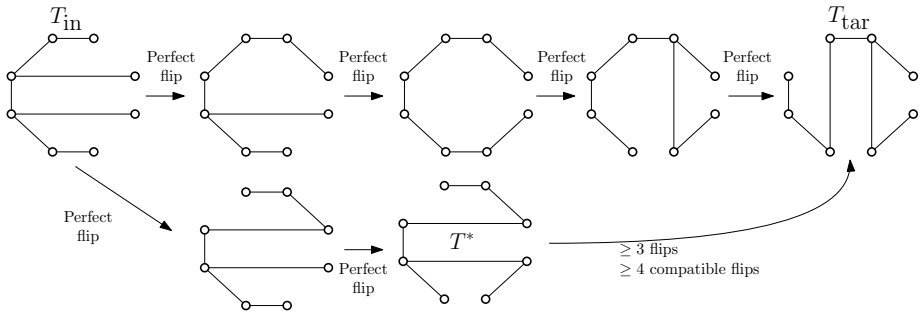


Figure 12 Counterexample to the perfect flip property based on [1, Figure 7]. The top shows the shortest flip sequence. The bottom sequence starts with two perfect flips, but reaches a point where no more perfect flips are possible. Since there are two edges in the tree T^* that both cross two edges from the target tree, at least three additional flips or four additional compatible flips are needed.

We remark that in [1, Claim 22] it is shown that for unrestricted flips Conjecture 19 implies Conjecture 17.

▶ **Lemma 20** ($\star$). *For any convex point set S and any two plane spanning trees T_{in} and T_{tar} on S, there is a shortest compatible flip sequence from T_{in} to T_{tar} that only uses parking edges from the boundary of the convex hull of S.*

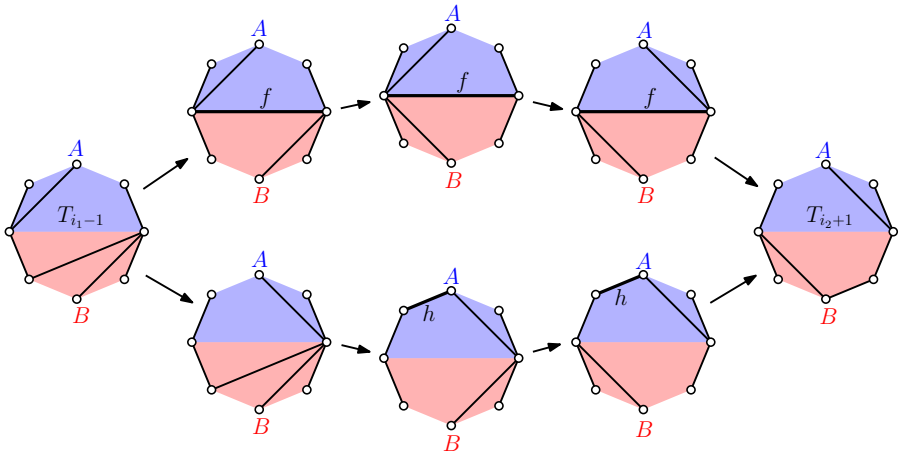


Figure 13 Reordering flips in Lemma 20. The top path is the part of the original flip sequence that contains f, the bottom path is the part of the reordered flip sequence that contains h.

Figure 13 shows the intuition behind the proof: One by one, we replace a parking edge f that is not on the boundary of the convex hull with a parking edge h on the convex hull boundary. To make this replacement possible, we change the order of the flips that happen while f is part of the tree. The edge f splits the convex point set into two sides, say A and B. Flips in either of the two sides can be executed independently from flips in the other side. Assume f is added when flipping from the tree T_{i_1-1} to T_{i_1} and removed in the flip from T_{i_2} to T_{i_2+1}. Exactly one of the sides, say side B, of T_{i_1-1} entirely contains a path that connects the two endpoints of f. The flips in the other side, say side A, can be executed before f gets added. Afterwards, we close a cycle in side A by adding the convex hull parking edge h and execute all the flips in side B. We conclude the new subsequence of flips by removing h and obtain the tree T_{i_2}.

We now show the strong happy edge property for compatible flips on trees.

▶ **Theorem 21.** *For any convex point set S and any two plane spanning trees T_{in} and T_{tar} on S, any compatible flip sequence from T_{in} to T_{tar} that removes (and adds) a happy edge is at least one step longer than the shortest compatible flip sequence from T_{in} to T_{tar}.*

Proof. Let $T_{\text{in}} = T_0, T_1,...,T_k = T_{\text{tar}}$ be a compatible flip sequence that removes a happy edge e in a flip from T_i to T_{i+1}. By Lemma 20, there exists a shortest compatible flip sequence from $T_{i+1} = T'_{i+1},T'_{i+2},...,T'_{\text{tar}} = T_{\text{tar}}$ that only uses parking edges from the convex hull boundary. Since e crosses neither an edge of $T_{\text{in}} \cap T_{\text{tar}}$ nor one on the convex hull boundary, e crosses no edge in the flip sequence $T_0,...,T_i,T'_{i+1},...,T_{\text{tar}}$. Hence, by Lemma 18, we can construct a shorter flip sequence that preserves e. ◀

By Proposition 16(i) and the example in Figure 12, Theorem 21 is best possible in the sense that there are flip sequences that flip a happy edge and cannot be shortened by two flips.

When we restrict the flip even further by only allowing edge rotations instead of all compatible exchanges, the happy edge property no longer holds. Also for the more restricted edge slides, the happy edge property does not hold, see [4] for the general case, and [1] for convex point sets.

Figure 14 shows a counterexample to the happy edge property for convex point sets with rotations. Assume the happy edge property would hold. Then the happy edge that passes through the middle of our convex polygon would split the tree into two proper subtrees. In each such subtree, the unhappy edge has a fixed position where it has to go. Since the initial position and the target position of the unhappy edge do not share any vertex, we need at least two rotations to get each unhappy edge from one position to the other. This gives us a minimum length of four for any flip sequence that preserves happy edges.

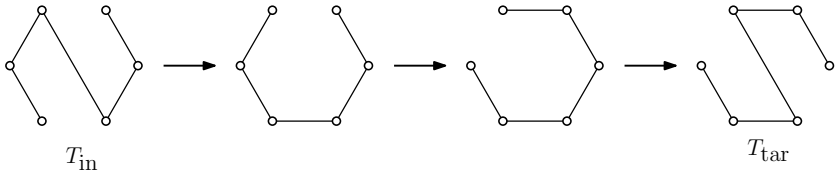


Figure 14 Counterexample to the happy edge property for the reconfiguration of plane spanning trees in convex point sets using rotations.

The flip sequence in Figure 14 only uses three rotations. Therefore, the happy edge property does not hold for the reconfiguration of plane spanning trees with rotations.

6 Fixed-Parameter Tractable Algorithms

Next, we show that the flip distance k and the compatible flip distance k_c are fixed-parameter tractable in k (resp. k_c). The core ingredient for all proofs is the happy edge property for convex hull edges, which is shown to hold for unrestricted flips in Proposition 18 in [1]. We obtain further improvements on the runtime built on the happy edge property for all edges and the parking edge property.

The upcoming lemma will describe a crucial step in the first two algorithms. Even if we never flip happy convex hull edges, the number of happy edges that are on the convex hull can still be arbitrarily large when compared to the number of unhappy edges. Therefore, we have to reduce that number in order to obtain a reasonable instance size that is bounded by

a function in k. We do this as follows: We mark all vertices that are incident to an unhappy edge that is in the initial configuration or the target configuration. All non-marked vertices are now only incident to convex hull edges. If we have a path between two marked vertices that only travels along convex hull edges incident to only non-marked vertices (except the first and/or last edge that start or end in a marked vertex), then we contract that path into one single edge (if both endpoints are marked) or a single vertex (if only one endpoint is marked). All the non-marked vertices get removed in the process. We call the new reduced instances $T_{\text{in,r}}$ and $T_{\text{tar,r}}$. The following Lemma states that in this contraction step no information about the length of the flip sequence is lost.

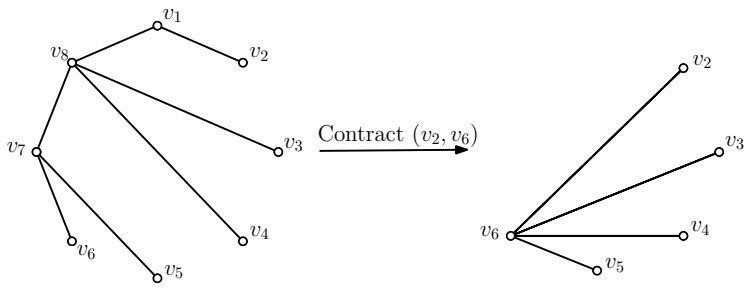


Figure 15 An example of a tree before (left) and after (right) contracting the path from v_2 to v_6 along the convex hull.

▶ **Lemma 22 (⋆).** *The (compatible) flip distance between T_{in} and T_{tar} has the same length as the flip distance between $T_{in,r}$ and $T_{tar,r}$.*

Now, we are ready to obtain our first FPT-algorithm. We remark that the notation O^* hides terms that are polynomial in n.

▶ **Theorem 23.** *The flip distance k for unrestricted flips is fixed-parameter tractable in k with runtime $O^*((ck^3)^k)$, where $c = 91.125$. The constant c can be improved to 8 if the happy edge property holds for unrestricted flips.*

Proof. For the unrestricted flip we define *good happy edges*. All happy edges on the convex hull are good happy edges. Further, a happy edge e that does not lie on the convex hull splits the point set into two parts. If one of the two parts contains only happy edges, then e is a good happy edge as well.

▷ Claim 24 (⋆). No good happy edge is flipped in any shortest flip sequence.

Now consider the tree $N(T_{\text{in}})$ that is obtained from T_{in} from exhaustively contracting happy leaves and paths of happy edges along the convex hull as in Lemma 22. We consider the dual tree of $N(T_{\text{in}})$, whose vertices are the faces of $N(T_{\text{in}}) \cup \{\text{convex hull edges}\}$ except the outer face and which contains edge between two vertices if and only if their corresponding faces share an edge of $N(T_{\text{in}})$.

▷ Claim 25 (⋆). Let P be a path in the dual tree that connects two faces containing distinct unhappy edges such that all the edges on traversed faces are happy edges. Then any shortest flip sequence either flips all those happy edges or none of them.

Let there be a total of t unhappy edges. Then there are a total of $t-1$ paths $P_1,\ldots,P_{t-1}$ that fit the form of Claim 25. Let $l_1,\ldots,l_{t-1}$ denote their number of happy edges in some order. Any flip sequence of length k gets to spend a total of at most $k-t$ flips on all these

happy edges combined. We enumerate all $2^{t-1} < 2^k$ subsets of $\{P_1, ..., P_{t-1}\}$ and if their number of happy edges sums up to less than $\frac{k-t}{2}$, then we consider an instance that contains all the faces that contain unhappy edges and the faces along the paths in the subset. Then the total remaining tree contains at most k unhappy edges from T_{in} and k unhappy edges from T_{tar}. That gives a total of at most $2k$ unhappy edges and a total of at most $4k$ vertices. Additionally, we have a total of less than $0.5k$ edges from faces with only happy edges. This gives another $0.5k$ vertices. So we have at most $4.5k$ vertices in our reduced forest. Brute forcing the optimal solution gives us a total of $4.5k - 1$ possible edges to remove and a total of at most $\binom{4.5k}{2}$ positions to add edges for every flip. This gives us a total runtime of $O^*\left(2^k \left(4.5k\binom{4.5k}{2}\right)^k\right) = O^*((91.125k^3)^k)$ based on the number of flip sequences that we have to check.

▷ Claim 26 (⋆). If Conjecture 17 is true, then the constant c in Theorem 23 improves to 8.

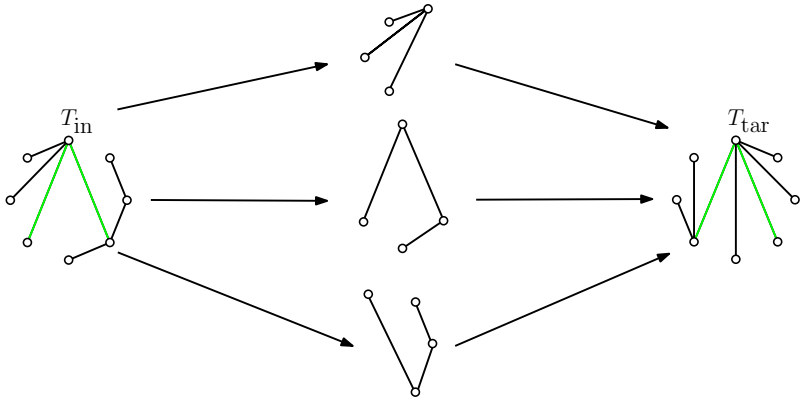


Figure 16 We split the initial tree T_{in} along its happy edges (colored in green) into three parts, from there we flip every part individually into its corresponding counterpart in T_{tar}.

Figure 16 gives an intuitive idea, where the improvement comes from. We cut the point set along all happy diagonals such that the new instances have all their happy edges on the convex hull. ◀

Next, we prove the existence of a faster FPT-algorithm for compatible flips that is based on the parking edge property.

▶ **Theorem 27.** *The compatible flip distance k_c is fixed-parameter tractable in k_c with runtime $O^*(((2+\epsilon)k_c^2)^{k_c})$, for any fixed $\epsilon > 0$.*

Proof. We divide the point set into components by cutting along happy edges and then find the shortest flip sequence for every remaining component individually, see Figure 16. Observe that all the happy edges in these components are convex hull edges. If all edges in a component are happy, do not flip any edge in it. By Lemma 20 there exists a shortest flip sequence that only uses edges from T_{in}, T_{tar} or the convex hull.

Now let $\epsilon > 0$ be fixed and $t = \lceil \frac{1}{\epsilon} \rceil$. Let C_1,...,C_m be all components with $\leq t$ unhappy edges. Determine the respective flip distances k_j from $T_{\text{in}}^{C_j}$ to $T_{\text{tar}}^{C_j}$ via brute force and discard all the instances. For checking whether the remaining instances have a combined flip distance of $k' = k_c - k_1 - ... - k_m$ one has to check $((2+\epsilon)k'^2)^{k'}$ different flip sequences. There are k' choices to remove an edge and every component with $l \geq n$ diagonals has $l + 1 \leq l(1 + \frac{1}{t}) \leq l(1 + \epsilon)$ holes in the convex hull where we can add edges. ◀

7 Conclusion

In this work, we have considered the compatible flip graph and the rotation graph, by improving the upper bound on the respective diameter from $2n - o(n)$ to $\frac{5}{3}(n-1)$ and to $\frac{7}{4}(n-1)$, respectively. The constructive upper bounds align with our findings about the happy edge property. On the one hand, we showed that any shortest compatible flip sequence preserves happy edges and that there is a shortest compatible flip sequence that only uses parking edges from the convex hull. The same holds for the compatible flip sequences that we used to obtain our upper bound. On the other hand, shortest rotation sequences may rotate happy edges. This may also happen in our upper bound construction. We further showed an immediate application of happy edge properties in trees, namely, that they can be used to derive fixed-parameter tractable algorithms. Open questions that are related to our work are:

1. Can we close the gap between upper and lower bounds for the flip distance, for each of the flip types? Do the diameters for different flip types differ or do they coincide?
2. Does the happy edge property still hold for flipping plane spanning trees if we drop one of the restrictions that either the point set is convex or the flips are all compatible?
3. What is the time complexity of finding shortest flip sequences for plane spanning trees? Again, this is interesting for each of the flip types.

References

1 Oswin Aichholzer, Brad Ballinger, Therese Biedl, Mirela Damian, Erik D. Demaine, Matias Korman, Anna Lubiw, Jayson Lynch, Josef Tkadlec, and Yushi Uno. Reconfiguration of non-crossing spanning trees. *Journal of Computational Geometry*, 15:224–253, 2024. doi:10.20382/jocg.v15i1a9.

2 Oswin Aichholzer, Joseph Dorfer, and Birgit Vogtenhuber. Constrained flips in plane spanning trees, 2025. arXiv:2508.15520.

3 Oswin Aichholzer, Wolfgang Mulzer, and Alexander Pilz. Flip distance between triangulations of a simple polygon is NP-complete. *Discrete & Computational Geometry*, 54(2):368–389, June 2015. doi:10.1007/s00454-015-9709-7.

4 Oswin Aichholzer and Klaus Reinhardt. A quadratic distance bound on sliding between crossing-free spanning trees. *Computational Geometry*, 37(3):155–161, 2007. Special Issue on the 20th European Workshop on Computational Geometry. doi:10.1016/j.comgeo.2004.12.010.

5 David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1):21–46, 1996. First International Colloquium on Graphs and Optimization. doi:10.1016/0166-218X(95)00026-N.

6 Håvard Bakke Bjerkevik, Linda Kleist, Torsten Ueckerdt, and Birgit Vogtenhuber. Flipping non-crossing spanning trees. *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2313–2325, 2025. doi:10.1137/1.9781611978322.77.

7 Miguel Bosch-Calvo and Steven Kelk. An improved kernel for the flip distance problem on simple convex polygons. *Inf. Process. Lett.*, 182(C), 2023. doi:10.1016/j.ipl.2023.106381.

8 Nicolas Bousquet, Lucas de Meyer, Théo Pierron, and Alexandra Wesolek. Reconfiguration of Plane Trees in Convex Geometric Graphs. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th International Symposium on Computational Geometry (SoCG 2024)*, volume 293 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:17, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2024.22.

9 Nicolas Bousquet, Lucas de Meyer, Théo Pierron, and Alexandra Wesolek. Reconfiguration of plane trees in convex geometric graphs. *CoRR*, 2023. doi:10.48550/arXiv.2310.18518.

10 Nicolas Bousquet, Valentin Gledel, Jonathan Narboni, and Théo Pierron. A note on the flip distance between non-crossing spanning trees. *Computing in Geometry and Topology*, 2023. doi:10.57717/cgt.v2i1.36.

11 Sean Cleary and Katherine St. John. Rotation distance is fixed-parameter tractable. *Inf. Process. Lett.*, 109(16):918–922, 2009. doi:10.1016/J.IPL.2009.04.023.

12 Carmen Hernando, Ferran Hurtado, Alberto Márquez, Mercè Mora, and Marc Noy. Geometric tree graphs of points in convex position. *Discrete Applied Mathematics*, 93(1):51–66, 1999. doi:10.1016/S0166-218X(99)00006-2.

13 M. Carmen Hernando, Ferran Hurtado, and Marc Noy. Graphs of non-crossing perfect matchings. *Graphs and Combinatorics*, 18:517–532, January 2002. doi:10.1007/s003730200038.

14 Ferran Hurtado. CCCG 2003; Paul Erdös Memorial Lecture: Flip, 2003.

15 Haohong Li and Ge Xia. An O (3.82^k) time FPT algorithm for convex flip distance. *Discrete & Computational Geometry*, pages 1–17, 2023.

16 Anna Lubiw and Vinayak Pathak. Flip distance between two triangulations of a point set is NP-complete. *Computational Geometry*, 49:17–23, 2015. doi:10.1016/j.comgeo.2014.11.001.

17 Joan Lucas. An improved kernel size for rotation distance in binary trees. *Information Processing Letters*, 110:481–484, June 2010. doi:10.1016/j.ipl.2010.04.022.

18 Torrie L. Nichols, Alexander Pilz, Csaba D. Tóth, and Ahad N. Zehmakan. Transition operations over plane trees. *Discrete Mathematics*, 343(8):111929, 2020. doi:10.1016/j.disc.2020.111929.

19 Alexander Pilz. Flip distance between triangulations of a planar point set is APX-hard. *Computational Geometry*, 47(5):589–604, 2014. doi:10.1016/j.comgeo.2014.01.001.

20 Daniel D. Sleator, Robert E. Tarjan, and William P. Thurston. Rotation distance, triangulations, and hyperbolic geometry. *Journal of the American Mathematical Society*, 1(3):647–681, July 1988. doi:10.1090/S0894-0347-1988-0928904-4.

Approximating Barnette's Conjecture

Michael A. Bekos
Department of Mathematics, University of Ioannina, Greece

Michael Kaufmann
Department of Computer Science, University of Tübingen, Germany

Maximilian Pfister
Department of Computer Science, University of Tübingen, Germany

Abstract

A well-known conjecture, named after David W. Barnette, asserts that every 3-regular, 3-connected, bipartite, planar graph (for short, Barnette graph) is Hamiltonian. As another step towards addressing Barnette's conjecture positively, we show that every n-vertex Barnette graph admits a subhamiltonian cycle containing $\frac{5n}{6}$ edges, improving upon the previous bound of $\frac{2n}{3}$. Equivalently, every Barnette graph admits a 2-page book embedding in which at least $\frac{5n}{6}$ consecutive vertex pairs along the spine are connected by edges. As a byproduct, we present a simple proof for a known result that guarantees the existence of Hamiltonian cycles in a certain subclass of Barnette graphs.

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

Keywords and phrases Barnette's Conjecture, Subhamiltonicity, Book embeddings

Digital Object Identifier 10.4230/LIPIcs.GD.2025.6

Funding *Michael A. Bekos*: Supported by the HFRI grant 26320.
Maximilian Pfister: Supported by the DFG grant SCHL 2331/1-1.

1 Introduction

Barnette's conjecture [3] is an intriguing open problem in topological graph theory; it asserts that every 3-regular, 3-connected, bipartite, planar graph (for short, Barnette graph) is Hamiltonian, i.e., it contains a simple cycle visiting each vertex exactly once. The conjecture is inspired by two earlier conjectures by Tait [12] and Tutte [13], both of which were disproven.

While Barnette's conjecture [3] remains open, several related results have been established. For instance, Goodey [7] showed that Barnette graphs with faces of degree 4 and 6 are Hamiltonian. Feder and Subi [5] extended this to Barnette graphs whose faces are 3-colored such that two colors contain only degree-4 and degree-6 faces. The authors in [2] proved Hamiltonicity under certain (improper) 2-colorings of the dual faces. Alternative forms of the conjecture have also been studied. Kelmans [11] showed that Barnette's conjecture is equivalent to the existence of a Hamiltonian cycle containing exactly one of any pair of edges on a common face. Other forms of equivalency include the existence of a Hamiltonian cycle through (i) any edge [8], (ii) any path of length 3 with two consecutive edges on the boundary of a face [9], and (iii) the middle edge of any path of length 3 avoiding its outer edges [9].

Using a computer-assisted method, Brinkmann, Goedgebeur and McKay [4] verified that all Barnette graphs with up to 90 vertices are Hamiltonian. Their proof relies on the fact that all Barnette graphs can be derived from the cube graph via cube- and C_4-expansions (see Figure 1) [10]. As the former operation preserves Hamiltonicity, the main challenge in Barnette's conjecture lies in handling the latter; see a related open problem at Section 5.

Our contribution. Since Hamiltonicity in a planar graph is equivalent to admitting a 2-page book embedding with all consecutive vertices (including the first and the last) connected, studying subhamiltonian cycles (that is, linear vertex orders in such embeddings) offers a

© Michael A. Bekos, Michael Kaufmann, and Maximilian Pfister;
licensed under Creative Commons License CC-BY 4.0

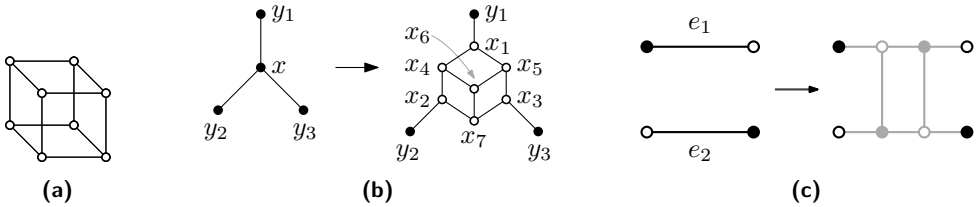


Figure 1 Starting from (a) the cube graph all Barnette graphs can be generated with either (b) cube-expansions or (c) C_4-expansions.

relaxed yet meaningful perspective. In this context, it is known that every n-vertex Barnette graph admits a subhamiltonian cycle containing at least $\frac{2n}{3}$ edges [1]. Building on this, we revisit and refine the approach of [1], improve the bound to $\frac{5n}{6}$, and discuss its potentials as a step toward resolving Barnette's conjecture. Note that a bound of n settles the conjecture.

2 Preliminaries

Basic definitions. We assume familiarity with basic concepts from Graph Theory and Graph Drawing. A k-page book embedding of a graph defines an order of its vertices and partitions its edges into k sets of non-crossing edges. A graph admitting a 2-page book embedding is called *subhamiltonian*, as it can be augmented to Hamiltonian by adding edges between consecutive vertices. The vertex order of such an embedding is called a *subhamiltonian cycle*.

Constructing Barnette graphs. As mentioned, all Barnette graphs can be constructed by iteratively applying two operations, cube-expansions and C_4-expansions, starting from the cube graph (see Figure 1). In a cube-expansion, a vertex x is replaced by seven new vertices $x_1, \ldots, x_7$ and the following edges: (x_1, y_1), (x_2, y_2), (x_3, y_3), (x_1, x_4), (x_1, x_5), (x_4, x_6), (x_5, x_6), (x_2, x_4), (x_2, x_7), (x_3, x_7), (x_6, x_7), and (x_3, x_5), where y_1, y_2, and y_3 are the neighbors of x. In a C_4-expansion, two edges e_1 and e_2 with an odd distance along the boundary of a face are each subdivided twice, and the resulting new vertices are connected with two additional edges by preserving both planarity and bipartiteness.

The algorithm by Alam et al. [1] Let G be an embedded Barnette graph with n vertices. By Lemma 2 of [1], the edges and the faces of G can be colored with three colors (red, blue and green) with the following properties (refer to Fig. 3(a) of [1] for an illustration): **(i)** every facial cycle of G is *bichromatic*, i.e., its edges alternate between two colors along its boundary, **(ii)** no two neighboring faces are of the same color, **(iii)** every face of G is colored differently from its bounding edges, **(iv)** the edge that bounds two faces of G colored x and y is of color z, where $\{x, y, z\} = \{\text{red}, \text{blue}, \text{green}\}$, **(v)** the subgraph G^*_{bg} of the dual G^* of G induced by the blue and green faces of G is connected. Let $\mathcal{T}^*_{bg}$ be a spanning tree of G^*_{bg}. This tree together with property (iv) yields a partition of the red edges of G into those that *are crossed by* $\mathcal{T}^*_{bg}$ (corresponding to those belonging to $\mathcal{T}^*_{bg}$) and the remaining ones that *are not crossed by* $\mathcal{T}^*_{bg}$. Then, the algorithm by Alam et al. [1] computes a subhamiltonian cycle C of G by a traversal of $\mathcal{T}^*_{bg}$, assuming that $\mathcal{T}^*_{bg}$ is rooted at an arbitrary blue leaf ρ as follows. For an arbitrary edge (u, v) of G that is crossed by $\mathcal{T}^*_{bg}$, let p and q be the faces that are bounded by (u, v), that is, (p, q) is an edge of $\mathcal{T}^*_{bg}$. The removal of (p, q) from $\mathcal{T}^*_{bg}$ results in two trees $\mathcal{T}^*_p$ and $\mathcal{T}^*_q$, such that w.l.o.g. ρ belongs to $\mathcal{T}^*_p$. For the recursive step, it is assumed that a *simple* and *plane* subhamiltonian cycle C_p for the subgraph G_p of G induced



(a) p is green, q blue **(b)** p is blue, q is green

Figure 2 The solid (dotted) gray edges belong to $\mathcal{T}^*_{bg}$ ($G^*_{bg} \setminus \mathcal{T}^*_{bg}$). The bold (plain) red edges are (are not, respectively) crossed by $\mathcal{T}^*_{bg}$. Cycle C_q is drawn dotted black. Figure reproduced from [1].

by the vertices of the faces of G in $\mathcal{T}^*_p$ has been computed, which does not necessarily span all vertices of G_p, but it satisfies the following invariant properties:

- **I.1** The edge (u, v) is on C_p.
- **I.2** Every red edge of G_p is in the interior of C_p or on C_p.
- **I.3** Every blue edge of G_p is in the exterior of C_p or on C_p.
- **I.4** Every green edge of G_p is on C_p.
- **I.5** For every red edge e of G_p that is not crossed by $\mathcal{T}^*_{bg}$ and that is adjacent to two faces h and h' of G, with $h \in \mathcal{T}^*_p$ and $h' \notin \mathcal{T}^*_p$, one of the following holds:
  - **a.** if h is a blue face, then both endpoints of e are on C_p,
  - **b.** if h is a green face, then none of the endpoints of e is on C_p.

Depending on the colors of faces p and q, the traversal of $\mathcal{T}^*_{bg}$ is continued by extending the cycle C_p to vertices of face q (thus obtaining a new cycle $\mathcal{C}_q$) and by maintaining Invariants I.1-I.5 as illustrated in Figure 2. Once the traversal of $\mathcal{T}^*_{bg}$ has been completed, a simple and plane cycle $\mathcal{C}$ has been computed, in which, by Invariants I.2-I.4, every green edge of G is on $\mathcal{C}$, while every red (blue) edge of G is in the interior (exterior) of cycle $\mathcal{C}$ or on $\mathcal{C}$. Since $\mathcal{T}^*_{bg}$ is a spanning tree of G^*_{bg}, every green edge of G bounds a face that is in $\mathcal{T}^*_{bg}$, and by Invariant I.4 both its endpoints are consecutive along $\mathcal{C}$. As every vertex is incident to a green edge, it follow that cycle $\mathcal{C}$ is indeed subhamiltonian. The following properties directly follow from Invariants I.4 and I.5, respectively (see [1]).

▶ **Property 1** ([1]). *Every green edge of G belongs to the subhamiltonian cycle $\mathcal{C}$.*

▶ **Property 2** ([1]). *Every red edge of G that is not crossed by $\mathcal{T}^*_{bg}$ belongs to the subhamiltonian cycle $\mathcal{C}$.*

The authors in [1] used Properties 1 and 2 to show that every Barnette graph admits a subhamiltonian cycle with $\frac{2n}{3}$ edges.

3 Our Main Result

To prove our main theorem, i.e., that every Barnette graph admits a subhamiltonian cycle containing at least $\frac{5n}{6}$ edges, we first establish a few more properties of the algorithm by Alam et al. [1]. To this end, let G be a Barnette graph and let f be a face of it. By Property (i) of the edge coloring of G, the edges on the boundary of f alternate between two colors. We say that two edges of the same color c are *consecutive* along f if and only if they appear consecutively in a clockwise traversal of the edges of color c in f.

▶ **Property 3.** *A blue edge appearing between two consecutive red edges of a green face of G that are crossed by $\mathcal{T}^*_{bg}$ belongs to the subhamiltonian cycle $\mathcal{C}$.*

Proof. Let f_g be a green face of G which contains a blue edge e_b appearing between two consecutive red edges e_r and e'_r of f_g that are crossed by $\mathcal{T}^*_{bg}$. Since e_r and e'_r are crossed by $\mathcal{T}^*_{bg}$, after the algorithm processes face f_g in the traversal of $\mathcal{T}^*_{bg}$, by Invariant I.1 both edges e_r and e'_r are on the subhamiltonian cycle constructed so far. By Invariant I.3, e_b is either in the exterior or on the subhamiltonian cycle constructed so far. Since e_b appears between e_r and e'_r along f_g, it follows that e_b cannot be in its exterior. Since the subhamiltonian cycle is not extended along blue edges (as those are not crossed by $\mathcal{T}^*_{bg}$), e_b will be part of $\mathcal{C}$. ◀

▶ **Property 4.** *Every green face of G that corresponds to a leaf of $\mathcal{T}^*_{bg}$ contributes a red edge (that is crossed by $\mathcal{T}^*_{bg}$) to the subhamiltonian cycle $\mathcal{C}$.*

Proof. Let f_g be a green face of G that corresponds to a leaf of $\mathcal{T}^*_{bg}$ and let f_b be its parent in $\mathcal{T}^*_{bg}$. Just before the algorithm processes face f_g in the traversal of $\mathcal{T}^*_{bg}$, the red edge of G that is crossed by the edge (f_g, f_b) of $\mathcal{T}^*_{bg}$ necessarily belongs to the subhamiltonian cycle constructed so far by Invariant I.1. Since f_g is a leaf of $\mathcal{T}^*_{bg}$ and since f_g is green, the subhamiltonian cycle will not be extended any further after the algorithm processes face f_g by Invariant I.5b. Hence, the red edge of G that is crossed by the edge (f_g, f_b) of $\mathcal{T}^*_{bg}$ will remain on the subhamiltonian cycle as desired. ◀

▶ **Property 5.** *Every green 4-face of G that does not correspond to a leaf of $\mathcal{T}^*_{bg}$ contributes two blue edges to the subhamiltonian cycle $\mathcal{C}$.*

Proof. Let f_g be a green 4-face (i.e., of degree 4) of G and assume that f_g does not correspond to a leaf of $\mathcal{T}^*_{bg}$. Since f_g is a 4-face of G, it consists of two red and two blue edges that alternate along its boundary. Since f_g does not correspond to a leaf of $\mathcal{T}^*_{bg}$, both red edges are crossed by $\mathcal{T}^*_{bg}$. Now the statement directly follows from Property 3. ◀

▶ **Property 6.** *Every green 6-face of G that does not correspond to a leaf of $\mathcal{T}^*_{bg}$ contributes at least one blue edge to the subhamiltonian cycle $\mathcal{C}$.*

Proof. Let f_g be a green 6-face of G that does not correspond to a leaf of $\mathcal{T}^*_{bg}$. Hence, its boundary contains at least two red edges that are crossed by $\mathcal{T}^*_{bg}$. These two red edges are consecutive on the boundary of f_g and thus the result follows by Property 3. ◀

So far, $\mathcal{T}^*_{bg}$ has only been assumed to be a spanning tree of G^*_{bg}. In the following, we show that it can be chosen in such a way that the preceding property generalizes to all green faces.

▶ **Property 7.** *There exists a choice of the tree $\mathcal{T}^*_{bg}$ such that every green face of G that does not correspond to a leaf of $\mathcal{T}^*_{bg}$ contributes at least one blue edge to the subhamiltonian cycle $\mathcal{C}$.*

Proof. Let f_g be a green face of G that is not a leaf in $\mathcal{T}^*_{bg}$. Since f_g is not a leaf, its boundary contains at least two red edges that are crossed by $\mathcal{T}^*_{bg}$. Our goal is to locally modify $\mathcal{T}^*_{bg}$ so that these two red edges appear consecutively along the boundary of f_g; Once this goal is achieved, we can apply Property 3 to obtain the desired result. Denote by $r_1, \ldots, r_k$ the red edges on the boundary of f_g in clockwise order. Let also $e_1, \ldots, e_k$ be the corresponding dual edges of G^*_{bg}. Finally, denote by v_g the vertex of G^*_{bg} corresponding to f_g. W.l.o.g. assume that r_1 is crossed by $\mathcal{T}^*_{bg}$, that is, e_1 belongs to $\mathcal{T}^*_{bg}$. Let r_j be the edge on the boundary of f that is crossed by $\mathcal{T}^*_{bg}$, such that $j > 1$ and j is minimum. Note that r_j exists, because f_g is not a leaf in $\mathcal{T}^*_{bg}$. If $j = 2$ or $j = k$ holds, then r_1 and r_2 are consecutive and we are done. Thus, assume $2 < j < k$. Consider the graph $\mathcal{T}^*_{bg} + e_2$. Since $\mathcal{T}^*_{bg}$ is spanning, $\mathcal{T}^*_{bg} + e_2$ contains a cycle. Let e be the edge of this cycle (distinct from e_2) that is also incident to v_g.

If $e \neq e_1$, then replacing e with e_2 in $\mathcal{T}_{bg}^*$ yields a new spanning tree in which e_1 and e_2 are consecutive, as desired. Hence, suppose $e = e_1$. In this case, we obtain a new spanning tree by replacing e_1 with e_2 in $\mathcal{T}_{bg}^*$ and we observe that the distance between the relevant pair of red edges crossed by this tree has been decreased. Repeating this argument a finite number of times will eventually yield a configuration in which the two red edges are consecutive, completing the proof. ◀

We are now ready to prove the main theorem of this section.

▶ **Theorem 1.** *Every n-vertex Barnette graph admits a subhamiltonian cycle with at least $\frac{5n}{6}$ edges.*

Proof. Let G be an n-vertex Barnette graph and let F_g, F_r and F_b be the set of green, red and blue faces of G, respectively. Since G is 3-regular, Euler's formula directly yields that G has (i) $\frac{3n}{2}$ edges out of which $\frac{n}{2}$ are green, $\frac{n}{2}$ are red and $\frac{n}{2}$ are blue, and (ii) $\frac{n}{2}+2$ faces. Hence, $|F_g| + |F_r| + |F_b| = \frac{n}{2} + 2$. Assume w.l.o.g. that $|F_g| \geq |F_r| \geq |F_b|$. By this assumption, F_g contains at least $\frac{1}{3}(\frac{n}{2}+2)$ faces. Thus, for some $x > 0$, we may write:

$$|F_g| = \frac{n}{6} + x \tag{1}$$

Since $|F_g| \geq |F_r| \geq |F_b|$, it follows:

$$|F_r| \geq \frac{n}{6} - \frac{x}{2} + 1 \tag{2}$$

We next claim that the average degree of the faces in F_g is at most 6. To see this, observe that edges on the boundary of the faces of F_g are in total n, since by Properties (iii) and (iv) of the edge coloring of G, they correspond to the red and the blue edges of G. Thus, by Eq.(1) the average degree of the faces in F_g is $\frac{n}{\frac{n}{6}+x}$, which is at most 6 as claimed. It follows that for every face in F_g of degree greater than 6, there is at least one face in F_g which has degree 4. With these observations in mind, we are now ready to count the number of edges contained in the subhamiltonian cycle $\mathcal{C}$.

- By Property 1, subhamiltonian cycle $\mathcal{C}$ contains all green edges of G, which are $\frac{n}{2}$ in total.
- By Property 2, subhamiltonian cycle $\mathcal{C}$ contains all red edges of G not crossed by $\mathcal{T}_{bg}^*$. We claim that these are at least $\frac{n}{6} - \frac{x}{2}$. To see this, recall that all red edges of G are $\frac{n}{2}$ in total. Since $\mathcal{T}_{bg}^*$ is a spanning tree of $\mathcal{G}_{bg}^*$, the number of red edges crossed by $\mathcal{T}_{bg}^*$ is:

  $$|F_b| + |F_g| - 1 = \frac{n}{2} + 2 - |F_r| - 1 = \frac{n}{2} - |F_r| + 1.$$

  Therefore, the number of red edges of G which are not crossed by $\mathcal{T}_{bg}^*$ is:

  $$\frac{n}{2} - (\frac{n}{2} - |F_r| + 1) = |F_r| - 1 \overset{(2)}{\geq} \frac{n}{6} - \frac{x}{2}$$

- Each green face of G contributes an additional edge to $\mathcal{C}$. To see this, observe that a green face corresponding to leaf in $\mathcal{T}_{bg}^*$ contributes at least one (red) edge to $\mathcal{C}$ by Property 4. Each of the remaining faces contributes at least one (blue) edge by Properties 5 and 7.

From the discussion above, it follows that the number of edges belonging to the subhamiltonian cycle $\mathcal{C}$ is at least

$$\frac{n}{2} + \frac{n}{6} - \frac{x}{2} + \frac{n}{6} + x = \frac{5n}{6} + \frac{x}{2}$$

which concludes the proof. ◀

4 Implications for Restricted Subclasses

In this section, we focus on certain restricted subclasses of Barnette graphs for which the algorithm yields a Hamiltonian cycle.

▶ **Observation 2.** *Let G be a Barnette graph. If $\mathcal{T}_{bg}^*$ is such that any green face f of G is (i) a leaf in $\mathcal{T}_{bg}^*$ or (ii) its degree in $\mathcal{T}_{bg}^*$ is exactly half of the degree of f, then G is Hamiltonian.*

Proof. A green leaf f contains exactly $\frac{1}{2}|f|$ red edges, one of which is crossed by $\mathcal{T}_{bg}^*$. The crossed edge will be part of $\mathcal{C}$ due to Property 4, while the others are part of $\mathcal{C}$ by construction. A green non-leaf f has, by assumption, maximum degree in $\mathcal{T}_{bg}^*$, i.e., all its red edges are crossed by $\mathcal{T}_{bg}^*$. But then Property 3 directly implies that $\frac{1}{2}|f|$ blue edges of f are part of $\mathcal{C}$. Hence, for any green face, half of its boundary edges are part of $\mathcal{C}$. Since the faces in F_g are disjoint and cover all red and blue edges, they include exactly n edges. Summing over all green faces, this contributes a total of $n/2$ edges to $\mathcal{C}$. Together with all green edges (which are $n/2$ many), this means that $|\mathcal{C}| = n$ and thus G is Hamiltonian. ◀

The following known result from the literature due to Florek [6] can hence be obtained as a direct corollary of Observation 2.

▶ **Corollary 3.** *A Barnette graph is Hamiltonian if its faces are 3-colored, with adjacent faces having different color, such that one of the three colors contain only faces of degree 4.*

Proof. W.l.o.g. let all green faces have degree 4. Hence, a green face is either a leaf in $\mathcal{T}_{bg}^*$ or its degree is 2 in $\mathcal{T}_{bg}^*$, i.e., maximum. Thus, Observation 2 yields the desired result. ◀

5 Open Problems

In this work, we approached Barnette's conjecture from the lens of 2-page book embeddings. Our work, while not resolving Barnette's conjecture at this time, makes another step forward and raises new open problems and possible directions for Barnette's conjecture:

1. Improve the lower bound of $\frac{5n}{6}$ on the size of the subhamiltonian cycle guaranteed by Theorem 1; a more strategic choice of the underlying spanning tree may lead to a better result in this direction (although our preliminary attempts were not successful).
2. It is also worth asking whether, for a given Barnette graph with a given Hamiltonian cycle, there is always a choice of the spanning tree $\mathcal{T}_{bg}^*$ such that our algorithm yields the given Hamiltonian cycle.
3. Is it true that all Barnette graphs generated solely by C_4-expansions are Hamiltonian?
4. The previous question is interesting also when the faces have a certain maximum degree.
5. Extend the set of restricted subclasses for which Observation 2 can be shown. Potential candidates are known subclasses from the literature which admit a Hamiltonian cycle.

References

1 Muhammad Jawaherul Alam, Michael A. Bekos, Vida Dujmović, Martin Gronemann, Michael Kaufmann, and Sergey Pupyrev. On dispersable book embeddings. *Theor. Comput. Sci.*, 861:1–22, 2021. doi:10.1016/J.TCS.2021.01.035.

2 Helmut Alt, Michael S. Payne, Jens M. Schmidt, and David R. Wood. Thoughts on Barnette's conjecture. *Australas. J Comb.*, 64:354–365, 2016. URL: http://ajc.maths.uq.edu.au/pdf/64/ajc_v64_p354.pdf.

3 David W. Barnette. Conjecture 5. In W. T. Tutte, editor, *Recent Progress in Combinatorics*. Academic Press, New York, 1969. Proc. of the Third Waterloo Conference on Combinatorics.

4 Gunnar Brinkmann, Jan Goedgebeur, and Brendan D. McKay. The minimality of the Georges-Kelmans graph. *Math. Comput.*, 91(335):1483–1500, 2022. doi:10.1090/MCOM/3701.

5 Tomas Feder and Carlos Subi. On Barnette's conjecture. Technical report, Electronic Colloquium on Computational Complexity (ECCC), 2006.

6 Jan Florek. Remarks on Barnette's conjecture. *J. Comb. Optim.*, 39(1):149–155, 2020. doi:10.1007/S10878-019-00460-8.

7 P. R. Goodey. Hamiltonian circuits in polytopes with even sided faces. *Israel Journal of Mathematics*, 22(1):52–56, March 1975. doi:10.1007/bf02757273.

8 A. Hertel. Hamiltonian cycles in sparse graphs. Master's thesis, University of Toronto, 2004.

9 A. Hertel. A survey and strengthening of Barnette's conjecture, 2005. Undergraduate thesis.

10 Derek A. Holton, Bennet Manvel, and Brendan D. McKay. Hamiltonian cycles in cubic 3-connected bipartite planar graphs. *J. Comb. Theory B*, 38(3):279–297, 1985. doi:10.1016/0095-8956(85)90072-3.

11 A. K. Kelmans. Konstruktsii kubicheskih dvudolnyh 3-svyaznyh bez gamiltonovyh tsiklov. *Sbornik Trudov Vserossiiskogo NII Sistemnykh Issledovanii*, 10:64–72, 1986.

12 P. G. Tait. Listing's topologie. *Philosophical Magazine*, 17:30–46, 1884. Reprinted in *Scientific Papers*, Vol. II, pp. 85–98.

13 W. T. Tutte. On hamiltonian circuits. *Journal of the London Mathematical Society*, 21(2):98–101, 1946. doi:10.1112/jlms/s1-21.2.98.

Same Quality Metrics, Different Graph Drawings

Simon van Wageningen
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Tamara Mchedlidze
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Alexandru C. Telea
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Abstract

Graph drawings are commonly used to visualize relational data. User understanding and performance are linked to the quality of such drawings, which is measured by quality metrics. The tacit knowledge in the graph drawing community about these quality metrics is that they are not always able to *accurately* capture the quality of graph drawings. In particular, such metrics may rate drawings with very poor quality as very good. In this work we make this tacit knowledge explicit by showing that we can modify existing graph drawings into arbitrary target shapes while keeping one or more quality metrics almost identical. This supports the claim that more advanced quality metrics are needed to capture the "goodness" of a graph drawing and that we cannot confidently rely on the value of a single (or several) certain quality metrics.

2012 ACM Subject Classification Human-centered computing → Graph drawings

Keywords and phrases graph drawing, quality metrics, assumptions, fooling

Digital Object Identifier 10.4230/LIPIcs.GD.2025.7

Related Version *Full Version*: `https://arxiv.org/abs/2508.15557`

Supplementary Material *Software (Source Code)*: `https://github.com/simonvw95/Same-Quality-Metric--Different-Graph-Drawings` [34]

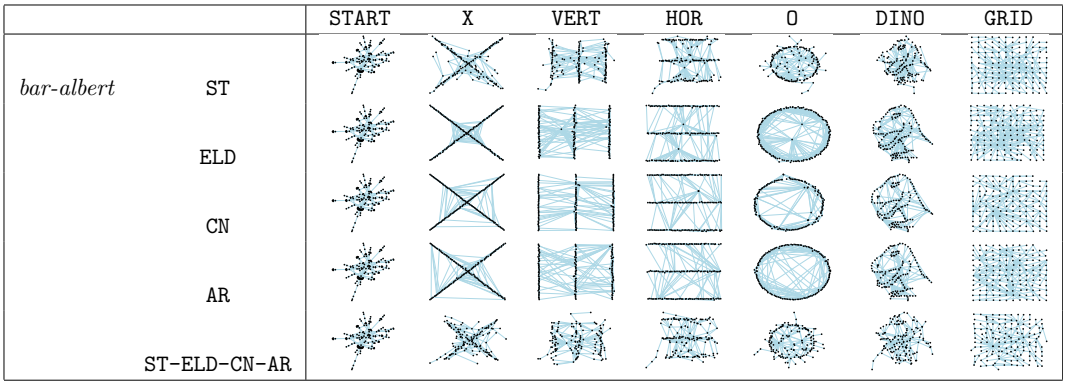


Figure 1 Morphing a graph drawing (`START`) into six different target shapes (`X,VERT,HOR,O,DINO,GRID`) while keeping one or more quality metric(s) nearly constant (`ST,ELD,CN,AR`). Values for `ST,ELD,AR` are within $\pm\epsilon = 0.0025$ and within $\pm\epsilon = \texttt{CN}(\Gamma) * 0.05$ for `CN`, so the morphed drawings have very similar quality to the starting ones.

1 Introduction

Relational data is typically visualized using node-link diagrams, commonly known as *graph drawings* (GD). High-quality graph drawings are key to user understanding and performance [31]. In turn, such quality is measured by various so-called *quality metrics* (QM) [30]. At

© Simon van Wageningen, Tamara Mchedlidze, and Alexandru C. Telea;
licensed under Creative Commons License CC-BY 4.0

a high level QMs capture two different aspects of the goodness of a GD: 1. how readable the elements of the drawing are, e.g., few crossings, few node overlaps – we call these *readability* metrics; and 2. how well the graph structure is represented by the drawing – *faithfulness metrics*. For instance, nodes lie at distances similar to their graph theoretical distances (measured by *stress*) or clusters are clearly depicted [28]

Quality metrics tend to correlate – often by design – with the aesthetic preferences of users [18, 31] and task performance [29]. As such, GD algorithms aim to (in)directly optimize them in order to produce pleasing drawings. For instance, spring-based techniques like Fruchterman-Reingold [15] and ForceAtlas2 [20] use forces on their nodes (and edges) in various ways, leading to drawings with "good" scores on most GD quality metrics. Other techniques like Stress Majorization [16], Stress-Plus-X [11], DeepGD [36], SGD^2 [1], and Core-GD [17] directly optimize one or more metrics to produce aesthetically pleasing drawings.

Two underlying and often implicit assumptions on quality metrics are that a "high" value indicates a "good" drawing; and that drawings of the same graph with similar metric values have similar quality. It is, however, widely known by Graph Drawing researchers, and has been exemplified multiple times by the Graph Drawing live challenge [12], that these assumptions do not always hold. In particular, over the years, GD challenge participants have presented drawings with very high QMs values that had an obfuscating appearance and did not reveal the graph structure.

We devote this work to make this tacit knowledge explicit and show that we cannot confidently rely on the value of a single quality metric nor on combinations of certain quality metrics. For this, we *"fool"* the quality metrics by morphing given, high-quality graph drawings into a given target shape (which can be any arbitrary shape) using a simulated annealing process while keeping one or more given quality metric values relatively fixed. Thus we are able to obtain drawings that obfuscate the graph structure and have very high values of quality metrics. While for readability metrics this is rather a confirmation of the tacit knowledge we already possess, it is very surprising to achieve this for a faithfulness quality metric such as stress as well as for combinations of certain QMs. We also show the limitations of our approach – certain combinations of metrics can not be fooled by it. Finally, we experiment with graphs of varying structures to illustrate how the graph's structure affects the difficulty of fooling quality metrics.

2 Related Work

The assumption that quality metrics directly correlate with human understanding, perception, and preference has been researched by many previous (user) studies [4]. The authors in [31] showed that larger numbers of edge crossings and edge bends are negatively correlated with human perception. Huang [18] refined such findings by using eye tracking to show that small edge angles lead to poorer performance in path search tasks.

The number of edge crossings in a drawing has been shown to correlate with user preferences [9, 31]. The perception of how well nodes are placed according to their shortest path, also called *stress*, was researched in a 2024 study [26], which found that users are able to perceive different values of stress in different drawings. Furthermore, stress also tends to correlate with user preferences [9].

Several types of visualizations beyond GD have been put to the test to try and fool statistical properties. Many works [2, 3, 7, 24] have shown that it is possible to control the visual appearance of 2D scatterplots, by iteratively morphing these into target shapes, while keeping summary statistical metrics such as means, standard deviations and correlation

similar. Different DR algorithms can create wildly different projections of the same dataset which have very similar quality metric values [14]. Separately, Machado, Telea, and Behrisch showed that one can control the shapes of clusters in projections and only marginally affect quality metrics [23], as well as completely distort the visual appearance of projections without affecting multiple quality metrics [22].

Similarly, Chari and Pachter [6] showed that they could transform projections of the same dataset into any specified shape without heavily adjusting the quality metric values. While such work did not aim to technically "fool" quality metrics, it provides strong evidence that metrics and projection appearance are not inherently correlated – an effect that we also show, in our work, to hold for graph drawings.

Regarding graphs (but not their drawings), Chen et al. [8] showed that one can generate graphs with the exact same graph statistics, such as the number of nodes, edges, and triangles. Di Bartolomeo, Lang and Dunne [13] (potentially a satirical paper) suggested as future research directions the idea of "rocketshipness", in which a graph drawing is optimized to resemble a rocket ship while maintaining one or more GD quality metrics. However, this idea was not further explored.

3 fooling GD metrics: Experimental setup

3.1 Preliminaries

We first introduce a few notations and concepts. An *undirected graph* $G = (V, E)$ is a set of nodes $V = \{v_1, \ldots, v_n\}$ and a set of undirected unweighted edges $E = \{e_1, \ldots, e_m\} \subseteq V \times V$. A straight-line *graph drawing* Γ of G assigns 2-dimensional coordinates X_i to nodes $v_i \in V$ and line segments L_i to edges $e_i \in E$, so Γ can be represented as a matrix $X^2 \in \mathbb{R}^{n\times 2}$ with rows $X_i \in \mathbb{R}^2$. Let $D \in \mathbb{R}^{n\times n}$ denote the *shortest path matrix* of graph-theoretic distances d_{ij} between all node-pairs $(v_i, v_j) \in V \times V$. We next refer to the Euclidean distance $\|X_i - X_j\|$ between two nodes v_i and v_j simply as *distance*. Let $deg(v)$ be the degree of node v, *i.e.*, the number of edges incident to v. A *quality metric* is a function $Q(\Gamma) \in [0, 1]$ which assigns a value to Γ, with low (resp. high) values denoting better (resp. poorer) drawings.

3.2 Metrics

We consider in our work the following GD quality metrics and their combinations:

Stress. Measures the difference between the Euclidean distances of all node-pairs and their shortest path distances [21] as $\texttt{ST}(\Gamma) = \frac{1}{n(n-1)/2} \sum_{i=0}^{n} \sum_{j=i+1}^{n} \frac{(\|Z_i - Z_j\| - d_{ij})^2}{d_{ij}^2}$, where Z_i scales the coordinate X_i by the shortest path distances as $Z_i = \frac{\sum_{i\neq j} \|X_i - X_j\|/d_{ij}}{\sum_{i\neq j} \|X_i - X_j\|^2/d_{ij}^2} X_i$.

$\texttt{ST}(\Gamma) = \frac{1}{n(n-1)/2} \sum_{i=0}^{n} \sum_{j=0}^{i} \frac{(\|Z_i - Z_j\| - d_{ij})^2}{d_{ij}^2}$,

We expect the ST metric to be the most difficult metric to fool in drawings, as it explicitly evaluates how well graph structure is depicted in a drawing, i.e., is a faithfulness QM.

Edge length deviation. Measures the average deviation of edge lengths [25] from the mean μ of all lengths in a drawing as $\texttt{ELD}(\Gamma) = \sqrt{\frac{1}{m} \sum_{i=1}^{m} \|L_i\| - \mu)^2}$. We expect this metric to be relatively easy to fool as it *solely* focuses on the uniformity of edge lengths.

Crossing number. We capture this by the number of intersections of all edge segments L_i, *i.e.* $\texttt{CN}(\Gamma) = \sum_{i=0}^{m} \sum_{j=i+1}^{m} \mathbf{1}(L_i \cap L_j \neq \emptyset)$. We expect that CN is hard to fool, as making subtle changes to any drawing tends to have a strong impact on how its lines cross.

Angular resolution. We measure the frequency of small angles occurrences in a graph drawing by computing the average deviation of angles [25] of adjacent edges *vs* the best possible angle as $\texttt{AR} = \frac{1}{n'} \sum_{i=1}^{n'} \left| \frac{\Theta_i - \theta_i}{\Theta_i} \right|$, where n' is the number of nodes with degree ≥ 2; θ_i is the smallest measured angle between consecutive adjacent edges of node i and $\Theta_i = 2\pi/deg(v_i)$ is the best possible angle between consecutive edges for node i. We speculate that this metric is easy to fool as it is easy to keep similar angles while making large visual changes.

3.3 Simulated Annealing Approach

We follow the approach in [24] to slowly morph any given graph drawing Γ with node coordinates X into a drawing Γ' with node coordinates X', while keeping one or more quality metrics values $QM^i(\Gamma')$, $i \in \{a, b, \dots\}$ very close to their original values $QM^i(\Gamma)$ and moving X' closer to a target shape Y. Here, Y can be *any* set of node coordinates, as long as $|Y| = |X|$, as this simplifies our morphing implementation.

Algorithm 1 Simulated Annealing Morphing.

```
1:  DIFF_curr ← Sim(X, Y)
2:  qm ← [QM^a(Γ), QM^b(Γ), ...]
3:  for i = 1 to N_max do
4:      (X', DIFF_test) ← Jit(X, T_i, DIFF_curr, Y)
5:      qm' ← [QM^a(Γ'), QM^b(Γ'), ...]
6:      if ∀j, |qm'[j] − qm[j]| ≤ ε_j then
7:          X ← X'
8:          DIFF_curr ← DIFF_test
9:      end if
10: end for
```

Algorithm 1 details our morphing: We start by computing the similarity of our graph drawing to the target shape and its quality metric value (lines 1-2). The *Jit* function jitters a random selection of nodes $X^s \subset X, 1 \leq |X|/15$ by a random value drawn from a normal distribution $[-0.5, 0.5]/25$ while increasing the similarity of X' to Y (line 4, see also Alg. 2). We accept X' if the quality $\mathbf{qm}'$ of its drawing Γ' is within a small range ϵ of the initial quality $\mathbf{qm}$ (lines 5-8). We repeat this process for $N_{max} = 30000$ iterations. To escape local minima, we also use simulated annealing: If a randomly generated value is below the current temperature T_i, we accept the node coordinates X_{jit}, irrespective of its (dis)similarity to Y. T_i varies over iterations: We start with $T_{init} = 0.4$ and quadratically decrease it to a final value $T_{N_{max}} = 0.001$. After exploring the practical ranges of the metrics in [25] and their variations discussed in [33], we set $\epsilon = 0.0025$ for ST,ELD,AR and $\epsilon = \texttt{CN}(\Gamma) * 0.05$ for CN.

For the similarity $Sim(X, Y)$ (see Alg. 3), one can use Mean Squared Error $MSE = \frac{1}{n} \sum_{i=1}^{n} (X_i - Y_i)^2$ or the Procrustes Statistic [37] (similar to *MSE* but invariant to rotation, translation and scaling). We also briefly experimented with the *Wasserstein Distance* [35] and the *Sinkhorn* [10] algorithm. However, these approaches yielded slightly worse results and were thus discarded.

Algorithm 2 Jitter function $Jit(X, T, DIFF, Y)$.

while do
  $X^s \subset X, 1 \leq |X|/15$
  $X_{jit} \leftarrow X + (X^s + \frac{\text{rand}(\mathcal{N}(-0.5,0.5))}{25})$
  if $Sim(X_{jit}, Y) < DIFF \vee T > \text{rand}(\mathcal{U}(0,1))$ **then**
    return $(X_{jit}, Sim(X_{jit}, Y))$
  end if
end while

The authors of [24] used the average distance of all points in X to the closest point in Y, which gives more morphing freedom. Our Sim design slightly adapts this idea. Algorithm 3 details our Similarity function: We start by computing the closest distances from two sets of node coordinates X and Y. We compute the distance of node X_0 (the *current* first node) in coordinate set X to all nodes in coordinate set Y. The distance to the closest node Y_{idx} in Y is then added to the total loss after which both X_0 and Y_{idx} are removed from X and Y, respectively. This iterative process is then repeated until all nodes are removed from the coordinate set and ensures no repetition of nodes in any pairs. To compare similarity values across different graphs, we express the similarity in a percentage, where we consider perfect similarity to the target shape ($Sim(Y, Y) = 0$) as 100% and perfect similarity to the starting shape as 0%, i.e.: $100 - (Sim(X', Y)/Sim(X, Y) * 100)$ where $Sim(X, Y) \rightarrow \mathbb{R}$ is equal to the starting similarity from Alg 1 (line 1).

Algorithm 3 Similarity function $Sim(X, Y)$.

$loss \leftarrow 0$
while $n > 0$ **do**
  $DIS \leftarrow [\|X_0 - Y\|]_{i=0}^{n}$
  $(idx, dis) \leftarrow (argmin(DIS), min(DIS))$
  $X \leftarrow RemoveRow(X, 0)$
  $Y \leftarrow RemoveRow(Y, idx)$
  $loss \leftarrow loss + dis$
  $n \leftarrow n - 1$
end while
return $loss$

3.4 Datasets

We perform experiments on the five graphs in Tab. 1. Four come from the Network Repository [32] due to their interesting varying structures. We created the fifth one using the

Table 1 Descriptive statistics of graphs used in our experiments.

Graph	n	m	$deg(V)_{avg}$	*source*
polbooks	105	441	8.40	[32]
lnsp_131	123	275	4.47	[32]
bar-albert	142	175	2.46	generated
gams10am	171	298	3.49	[32]
dwt_307	307	1108	7.22	[32]

dual-Barabasi-Albert algorithm [27] to control the node count n. For each graph we create initial drawings (START) using ForceAtlas2 [20] with 5000 iterations and default parameters. We translate and scale all drawings to have coordinates in $[0, 1]$ so that *Jit* becomes scale invariant. We use six target shapes Y from [24] including the Dinosaur shape from [5], see Fig. 2. For all shapes we create simple algorithms that generate Y for any point count n so we can next easily enforce $|X| = |Y|$ (see Sec. 3.3). Our data and code are available on GitHub [34].

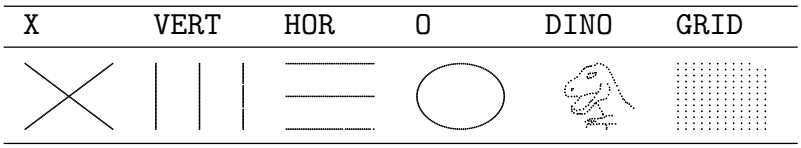


Figure 2 Target shapes used in our experiments.

4 Results and Discussion

We next refer to results from the figures and tables in the Appendix with the ⋆ notation. For larger versions of these figures check the full paper version. These extra results include the drawings of all pairs and triples of metric combinations, distribution of similarity percentages, and results of statistical testing. We also refer the reader to our video material, in which we replicate a dancing person while keeping a quality metric value the same. An example of four frames of this video can be seen in Fig. 3. Additional videos and details on how the videos were created can be found on our GitHub page [34].

Individual metrics. Figures 1 and 4 show the resulting drawings of fooling individual metrics and all four metrics at once. We see that not all individual quality metrics can be fooled equally well. As speculated, fooling ST proves to be the most difficult. For example, the *bar-albert* graph nearly perfectly morphs to all six targets for ELD,CN,AR but not for ST. We see similar results for *polbooks*, *lnsp_131* and *dwt_307*. For the metrics ELD and AR, we observe easy fooling for all graphs except *dwt_307*. We also see that fooling the number of crossings (CN) is easier than expected – see the results for all graphs except *gams01am* and *dwt_307*. Aggregating the results over all graphs and target shapes (see Fig. 7⋆) shows that ELD is much easier to fool than ST; and AR is much easier to fool than ST and CN.

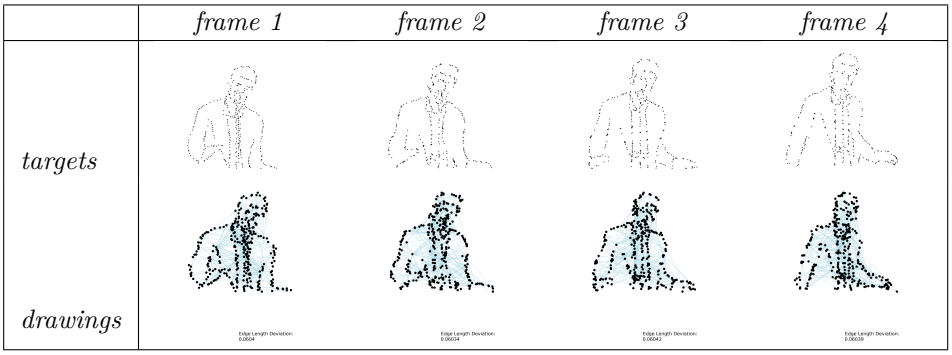


Figure 3 Four sequential target frames from the video and the resulting sequential drawings from attempting to fool a quality metric (ELD) for the target frames.

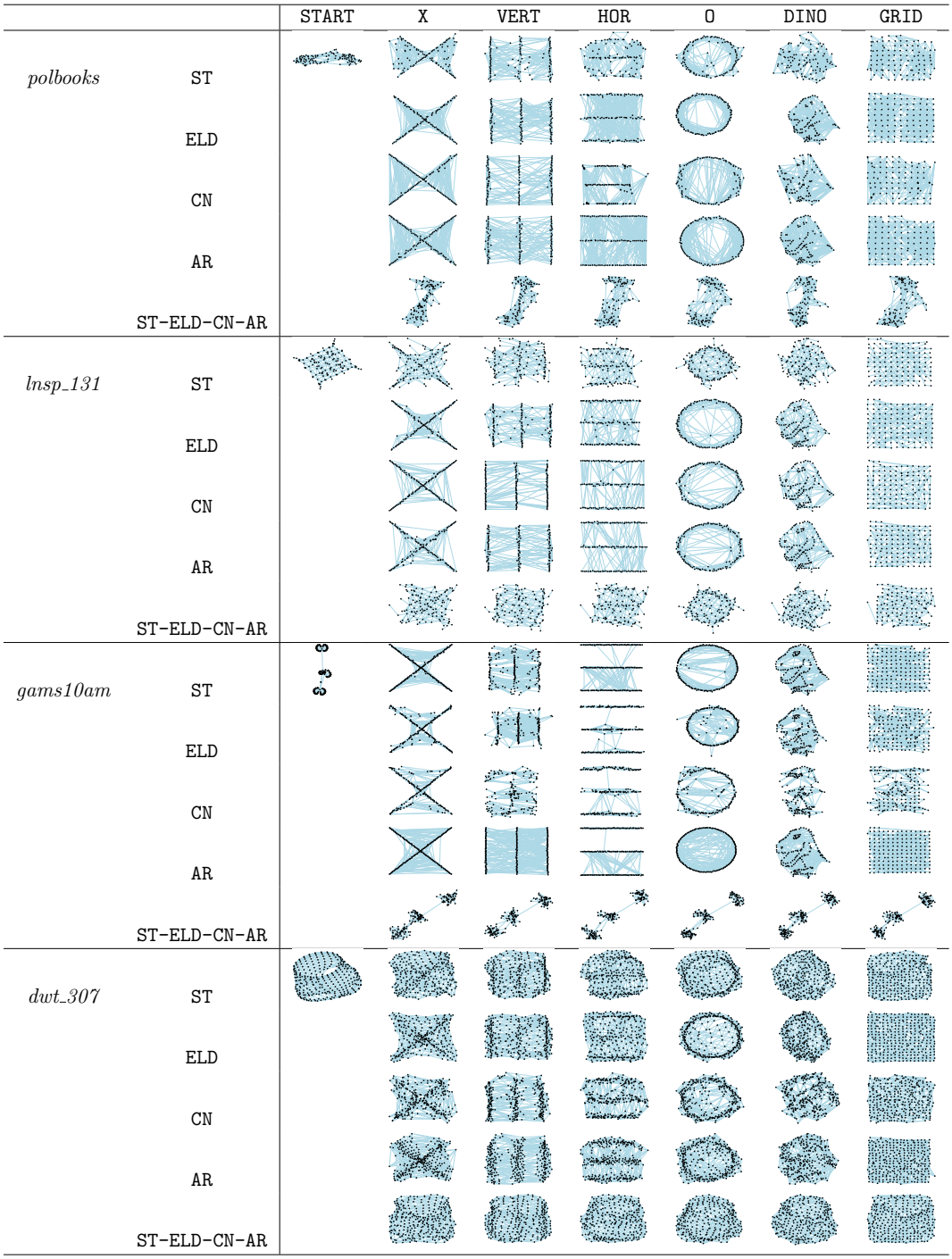


Figure 4 Collection of different graph drawings. The `START` column indicates the starting drawing of various graphs. The results of applying Alg. 1 to `START` to six target shapes (`X`, `VERT`, `HOR`, `O`, `DINO`, `GRID`) are shown in their respective columns. The rows indicate the (combinations of) metrics that have $\pm\epsilon = 0.0025$ for `ST,ELD,AR` and $\pm\epsilon = \texttt{CN}(\Gamma) * 0.05$ for `CN`.

Combination of metrics. We observe that our technique is almost never able to fool all four metrics at once, except for the *bar-albert* graph. Looking at the drawings in Fig. 5⋆ and the similarity values in Fig. 6⋆, we see that fooling three metrics at the same time is also hard. In contrast, fooling combinations of two metrics is sometimes possible but proves to be hard for *dwt_307* and *gam10am*. We observe that the combination `ST-CN` is rather resistant to fooling; every other metric pair is much easier to fool than `ST-CN` except for `ELD-CN`.

Graph type. The spread of the similarity values in Fig. 6⋆, and in particular the low values for *dwt_307* and *bar-albert*, support our hypothesis that the graph *type* can influence how difficult it is to fool its quality metrics. For *dwt_307*, we see that the target shapes (except `GRID`) are much less visible for most metrics compared to results of other graphs. In Fig. 8⋆, we see that the metrics and their combinations for this graph are significantly harder to fool than other graphs, most likely due to its strong mesh-like structure. The metrics and their combinations for the *bar-albert* graph, whose tree-like structure allows for a lot of layout flexibility, are significantly easier to fool compared to all other graphs in our dataset.

Target shape. Lastly, we observe that some target *shapes* are harder to morph to. For instance, `DINO` is much more complex than `O` or `X`. This leads to messy drawings for some graphs – *dwt_307* is unable to even come close. When aggregating the results over all graphs and metric combinations (see Fig. 8⋆), we see that only the `GRID` shape is significantly easier to morph to than all other shapes.

5 Conclusion

In this work we provide evidence against the assumptions that (1) "high" quality metric values indicate "good" graph drawings; and that (2) drawings of the same graph with similar metric values are similar in their quality. We use simulated annealing to morph existing graph drawings into six different, arbitrary, target shapes without substantially altering one or more quality metric values. To our knowledge, this is the first time that a systematic fooling of graph drawing quality metrics has been explored. Depending on the graph and metric, most graph drawings can easily be morphed towards such targets. We observe that the structure of the graph has an influence on how well a graph drawing can fooled. Furthermore, we find evidence that some metrics (*stress*) are more difficult to fool than others, depending on the target shape; and that some combinations of metrics are easier to fool. All in all, our results suggest that current quality metrics are not enough to capture the essence of a "good" graph drawing on their own. Similar to Huang et al. [19], we emphasize the importance of combinations of two or more quality metrics that capture both the graph structure (faithfulness) and readability of the drawing. Future work can consider advancing our fooling method by using more sophisticated techniques such as gradient descent or deep learning approaches. Furthermore, more experiments can be conducted with more graphs, target drawings, and quality metrics, as well as linear combinations of multiple quality metrics. Lastly, a perceptual user-study could explore how readable or misleading these morphed drawings are.

References

1 Reyan Ahmed, Felice De Luca, Sabin Devkota, Stephen Kobourov, and Mingwei Li. Multicriteria Scalable Graph Drawing via Stochastic Gradient Descent, $(SGD)^2$. *IEEE TVCG*, 28(06):2388–2399, 2022. doi:10.1109/TVCG.2022.3155564.

2 Francis J. Anscombe. Graphs in Statistical Analysis. *The American Statistician*, 27(1):17–21, 1973. doi:10.1080/00031305.1973.10478966.

3 Tal Boger, Steven B. Most, and Steven L. Franconeri. Jurassic Mark: Inattentional Blindness for a Datasaurus Reveals that Visualizations are Explored, not Seen. In *IEEE Vis*, pages 71–75, 2021. doi:10.1109/VIS49827.2021.9623273.

4 Michael Burch, Weidong Huang, Mathew Wakefield, Helen C. Purchase, Daniel Weiskopf, and Jie Hua. The State of the Art in Empirical User Evaluation of Graph Visualizations. *IEEE Access*, 9:4173–4198, 2021. doi:10.1109/ACCESS.2020.3047616.

5 Albert Cairo. Download the Datasaurus: Never trust summary statistics alone; always visualize your data, 2016. URL: https://thefunctionalart.blogspot.com/2016/08/download-datasaurus-never-trust-summary.html.

6 Tara Chari and Lior Pachter. The specious art of single-cell genomics. *Computational Biology*, 19(8):1–20, 2023. doi:10.1371/journal.pcbi.1011288.

7 Sangit Chatterjee and Aykut Firat. Generating Data with Identical Statistics but Dissimilar Graphics. *The American Statistician*, 61(3):248–254, 2007. doi:10.1198/000313007X220057.

8 Hang Chen, Utkarsh Soni, Yafeng Lu, Vahan Huroyan, Ross Maciejewski, and Stephen G. Kobourov. Same Stats, Different Graphs: Exploring the Space of Graphs in Terms of Graph Properties . *IEEE TVCG*, 27(03):2056–2072, 2021. doi:10.1109/TVCG.2019.2946558.

9 Markus Chimani, Patrick Eades, Peter Eades, Seok-Hee Hong, Weidong Huang, Karsten Klein, Michael R. Marner, Ross T. Smith, and Bruce H. Thomas. People prefer less stress and fewer crossings. In *Graph Drawing*, volume 8871 of *LNCS*, pages 523–524. Springer Berlin Heidelberg, 2014.

10 Marco Cuturi. Sinkhorn distances: lightspeed computation of optimal transport. In *NIPS*, volume 2, pages 2292–2300. Curran Associates Inc., 2013. doi:10.48550/arXiv.1306.0895.

11 Sabin Devkota, Reyan Ahmed, Felice De Luca, Katherine E. Isaacs, and Stephen G. Kobourov. Stress-Plus-X (SPX) Graph Layout. In *Graph Drawing*, volume 11904 of *LNCS*, pages 291–304. Springer Berlin Heidelberg, 2019. doi:10.1007/978-3-030-35802-0_23.

12 Sara Di Bartolomeo, Fabian Klute, Debajyoti Mondal, and Jules Wulms. Graph drawing contest report. In *Graph Drawing*, LIPIcs, pages 41:1–41:13. Schloss Dagstuhl, October 2024. doi:10.4230/LIPICS.GD.2024.41.

13 Sara Di Bartolomeo, Matěj Lang, and Cody Dunne. The worst graph layout algorithm ever. In *Proc. alt. VIS workshop at IEEE VIS*, 2022. doi:10.31219/osf.io/4hfy9.

14 Mateus Espadoto, Rafael M. Martins, Andreas Kerren, Nina S. T. Hirata, and Alexandru C. Telea. Toward a Quantitative Survey of Dimension Reduction Techniques. *IEEE TVCG*, 27(3):2153–2173, 2919. doi:10.1109/TVCG.2019.2944182.

15 Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991. doi:10.1002/spe.4380211102.

16 Emden R. Gansner, Yehuda Koren, and Stephen C. North. Graph Drawing by Stress Majorization. In *Graph Drawing*, volume 3383 of *LNCS*, pages 239–250. Springer Berlin Heidelberg, 2005. doi:10.1007/978-3-540-31843-9_25.

17 Florian Grötschla, Joël Mathys, Robert Veres, and Roger Wattenhofer. Core-GD: A hierarchical framework for scalable graph visualization with GNNs. In *The Twelfth International Conference on Learning Representations*, 2024. doi:10.48550/arXiv.2402.06706.

18 Weidong Huang. Using eye tracking to investigate graph layout effects. In *APVIS*, pages 97–100. IEEE Computer Society, 2007. doi:10.1109/APVIS.2007.329282.

19 Weidong Huang, Peter Eades, Seok-Hee Hong, and Chun-Cheng Lin. Improving multiple aesthetics produces better graph drawings. *Journal of Visual Languages and Computing*, 24(4):262–272, 2013. doi:10.1016/j.jvlc.2011.12.002.

20 Mathieu Jacomy, Tommaso Venturini, Sebastien Heymann, and Mathieu Bastian. ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software. *PLOS ONE*, 9(6):e98679, 2014. doi:10.1371/journal.pone.0098679.

21 Tomihisa Kamada and Satoru Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989. doi:10.1016/0020-0190(89)90102-6.

22 Alister Machado, Michael Behrisch, and Alexandru C. Telea. Necessary but not sufficient: Limitations of projection quality metrics. *Computer Graphics Forum*, page e70101, 2015. doi:10.1111/cgf.70101.

23 Alister Machado, Alexandru C. Telea, and Michael Behrisch. ShaRP: Shape-Regularized Multidimensional Projections. In *EuroVis Workshop on Visual Analytics (EuroVA)*. The Eurographics Association, 2023. doi:10.2312/eurova.20231088.

24 Justin Matejka and George Fitzmaurice. Same Stats, Different Graphs: Generating Datasets with Varied Appearance and Identical Statistics through Simulated Annealing. In *CHI*, pages 1290–1294. Association for Computing Machinery, 2017. doi:10.1145/3025453.3025912.

25 Gavin J. Mooney, Helen C. Purchase, Michael Wybrow, and Stephen G. Kobourov. The Multi-Dimensional Landscape of Graph Drawing Metrics. In *IEEE PacificVis*, pages 122–131, 2024. doi:10.1109/PacificVis60374.2024.00022.

26 Gavin J. Mooney, Helen C. Purchase, Michael Wybrow, Stephen G. Kobourov, and Jacob Miller. The Perception of Stress in Graph Drawings. In *Graph Drawing*, volume 320 of *LIPIcs*, pages 21:1–21:17. Schloss Dagstuhl, 2024. doi:10.4230/LIPIcs.GD.2024.21.

27 Niema Moshiri. The dual-Barabási-Albert model. *arXiv preprint arXiv:1810.10538*, 2018.

28 Quan Nguyen, Peter Eades, and Seok-Hee Hong. On the faithfulness of graph visualizations. In *2013 IEEE PacificVis*, pages 209–216, 2013. doi:10.1109/PacificVis.2013.6596147.

29 Helen C. Purchase. Which aesthetic has the greatest effect on human understanding? In *Graph Drawing*, volume 1353 of *LNCS*, pages 248–261. Springer Berlin Heidelberg, 1997. doi:10.1007/3-540-63938-1_67.

30 Helen C. Purchase. Metrics for Graph Drawing Aesthetics. *Journal of Visual Languages and Computing*, 13(5):501–516, 2002. doi:10.1006/jvlc.2002.0232.

31 Helen C. Purchase, Robert F. Cohen, and Murray James. Validating graph drawing aesthetics. In *Graph Drawing*, volume 1027 of *LNCS*, pages 435–446. Springer Berlin Heidelberg, 1996. doi:10.1007/bfb0021827.

32 Ryan A. Rossi and Nesreen K. Ahmed. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*, 2015. URL: https://networkrepository.com.

33 Simon van Wageningen, Tamara Mchedlidze, and Alexandru C. Telea. An Experimental Evaluation of Viewpoint-Based 3D Graph Drawing. *Computer Graphics Forum*, 43(3):e15077, 2024. doi:10.1111/cgf.15077.

34 Simon van Wageningen, Tamara Mchedlidze, and Alexandru C. Telea. Same Quality Metrics source code and videos, 2025. URL: https://github.com/simonvw95/Same-Quality-Metric--Different-Graph-Drawings.

35 Cédric Villani. *The Wasserstein distances*, pages 93–111. Springer, 2009. doi:10.1007/978-3-540-71050-9_6.

36 Xiaoqi Wang, Kevin Yen, Yifan Hu, and Han-Wei Shen. DeepGD: A deep learning framework for graph drawing using GNN. *IEEE CGA*, 41(05):32–44, 2021. doi:10.1109/MCG.2021.3093908.

37 Yong Wang, Zhihua Jin, Qianwen Wang, Weiwei Cui, Tengfei Ma, and Huamin Qu. DeepDrawing: A Deep Learning Approach to Graph Drawing. *IEEE TVCG*, pages 1–1, 2019. doi:10.1109/TVCG.2019.2934798.

A Appendix

A.1 Combinations of Metrics

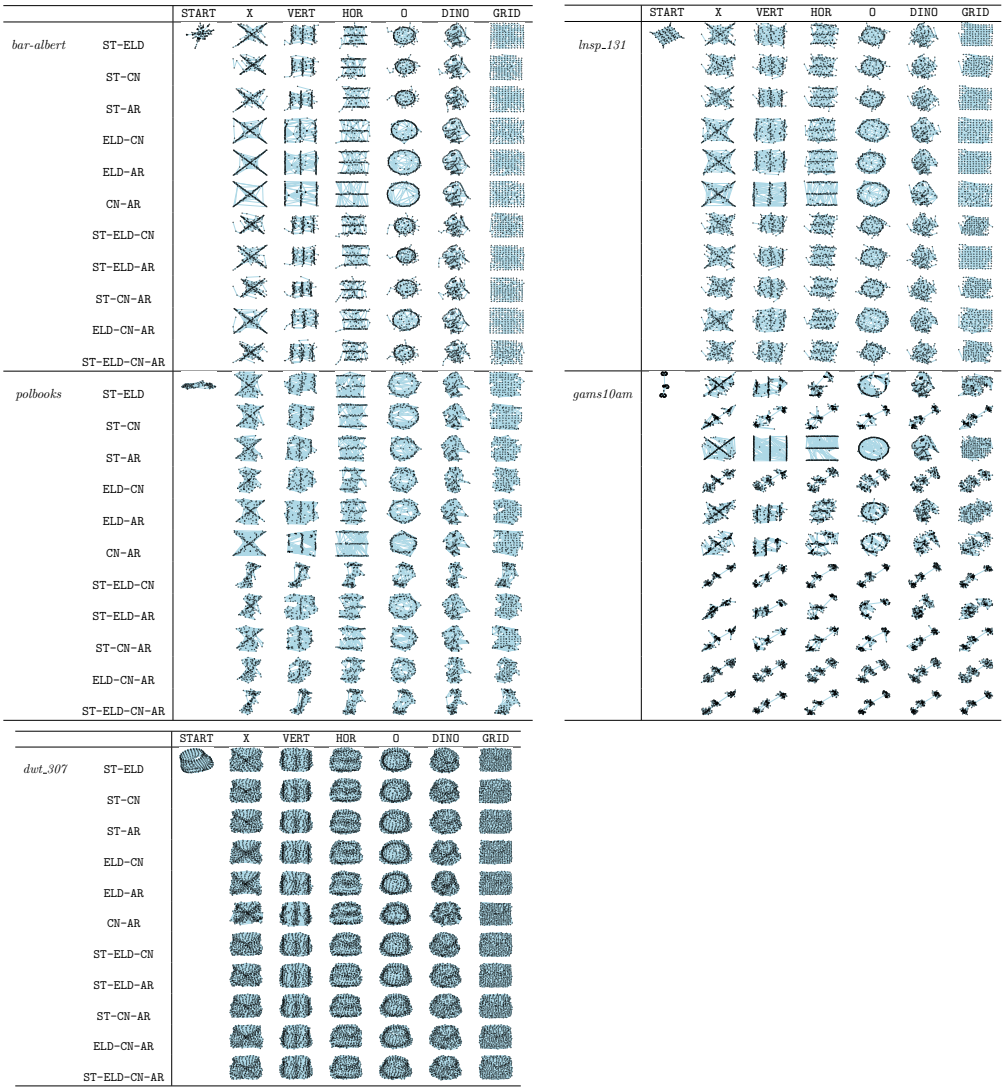


Figure 5 Collection of different graph drawings. The `START` column indicates the starting drawing of various graphs. The results of applying Alg. 1 to `START` to six target shapes (`X`, `VERT`, `HOR`, `O`, `DINO`, `GRID`) are shown in their respective columns. The rows indicate the metrics that have $\pm\epsilon = 0.0025$ for combinations of `ST,ELD,AR` and $\pm\epsilon = \texttt{CN}(\Gamma) * 0.05$ for `CN`.

A.2 Similarity percentages

Figure 6⋆ visualizes the distributions of the similarity percentages. Here, each dot represents how similar the fooled drawing is to its target shape.

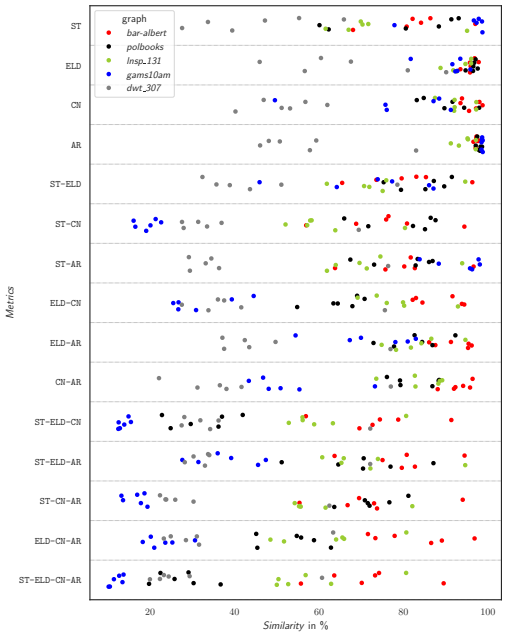


Figure 6 Distributions of how similar the resulting drawings are to the six target shapes.

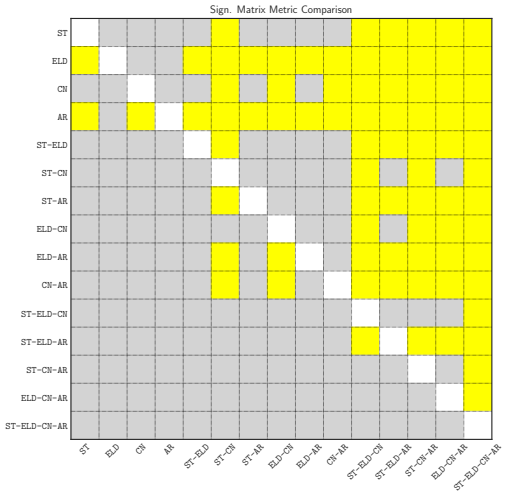


Figure 7 Comparing the distributions of each metric's results with each other metric. Each box indicates that the resulting distribution of the results of the row metric minus the results of the column metric are non-significantly (gray) and significantly (yellow) greater (easier to fool) than a distribution symmetric about zero.

A.2.1 Metric comparison

To test significant differences between the non-normally distributed results of the fooled metrics we first employ the Friedman test. We find that there are significant differences ($\chi^2 = 314$, $p < 0.05$) between the results of metrics. Since our data is paired, we then perform pairwise metric comparisons using the Wilcoxon signed-rank test. Additionally, we correct the p-values using the Bonferroni adjustments.

In Figure 7⋆ we display the results of the significance tests with $p < 0.05$. Here, the metric of a row is compared with the metrics of all columns and vice-versa. A yellow (resp., gray) color indicates that the resulting distribution of the row metric minus the column metric is significantly (resp., or non-significantly) greater than a distribution symmetric about zero. As we are using a *greater than* alternative hypothesis, the pattern in Figure 7⋆ is not symmetric. For example, for `ELD` and `ST` we see that there is a significant difference, meaning that the difference between the distributions of `ELD` and `ST` is greater than a distribution around zero, i.e. `ELD` is easier to fool than `ST`. However, that means that the difference between the distributions of `ST` and `ELD` is smaller than a distribution around zero which is why this pair is not significant for the "greater than" test.

From this pair-wise metric comparison, we observe that fooling individual metrics is easier than fooling combinations, with `ELD` and `AR` being the simplest. For pairs of quality metrics, the `ST-CN` combination is more resistant to fooling than others. We see the same for any other larger combination that includes `ST` and `CN`, such as `ST-ELD-CN` and `ST-CN-AR` which are the most difficult to fool.

A.2.2 Target shape comparison

We test the differences between graphs in exactly the same manner as for the metric comparison, with now the dependent variable being the target shape. The Friedman test shows that there are significant differences between the results of the target shapes ($\chi^2 = 42.9$, $p < 0.05$). The results of the pairwise Wilcoxon signed-rank tests are visualized in Figure 8⋆. From these results we observe that the `GRID` shape is significantly easier to morph into than any other shape. All the other shapes are more or less equally difficult to morph any drawing into, with the exception that the `DINO` shape is significantly easier than `VERT`.

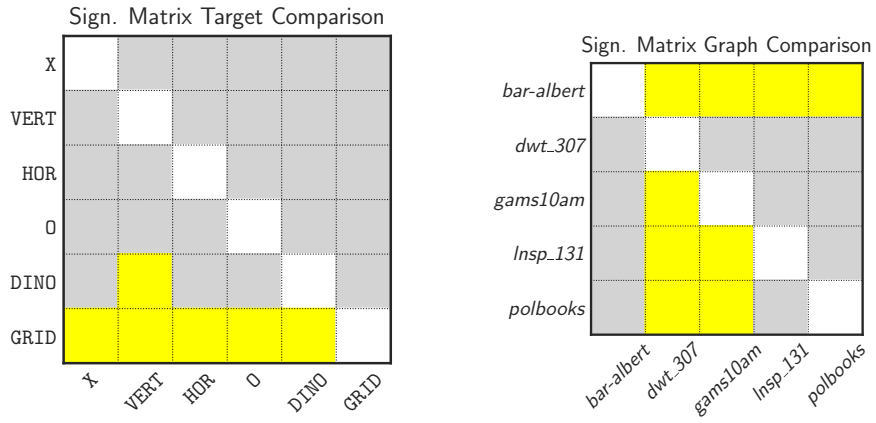


Figure 8 Comparing the distributions of each target's results with each other target (left) and comparing each graph with each other graph (right). Each box indicates that the resulting distribution of the results of the row metric target the results of the column target are non-significantly (gray) and significantly (yellow) greater (easier to fool) than a distribution symmetric about zero.

A.2.3 Graph comparison

We test the differences between graphs in a similar way to the metric comparison, with now the dependent variable being the graph. However, due to the graph data being independent samples we use the Mann-Whitney U test to determine significant differences between pairs of graphs. These results are visualized in Figure 8⋆. From these results we observe that for the *bar-albert* graph it significantly easier to fool any metric combination than it is for any other graph. Conversely, fooling metrics for the *dwt_307* graph is significantly more difficult than for any other graph

Tangling and Untangling Trees on Point-Sets

Giuseppe Di Battista
Roma Tre University, Italy

Giuseppe Liotta
University of Perugia, Italy

Maurizio Patrignani
Roma Tre University, Italy

Antonios Symvonis
School of Applied Mathematical & Physical Sciences, National Technical University of Athens, Greece

Ioannis G. Tollis
Department of Computer Science, University of Crete, Heraklion, Greece

Abstract

We study a question that lies at the intersection of classical research subjects in Topological Graph Theory and Graph Drawing: Computing a drawing of a graph with a prescribed number of crossings on a given set S of points, while ensuring that its curve complexity (i.e., maximum number of bends per edge) is bounded by a constant. We focus on trees: Let T be a tree, $\vartheta(T)$ be its thrackle number, and χ be any integer in the interval $[0, \vartheta(T)]$. In the *tangling* phase we compute a topological linear embedding of T with $\vartheta(T)$ edge crossings and a constant number of spine traversals. In the *untangling* phase we remove edge crossings without increasing the spine traversals until we reach χ crossings. The computed linear embedding is used to construct a drawing of T on S with χ crossings and constant curve complexity. Our approach gives rise to an $O(n^2)$-time algorithm for general trees and an $O(n \log n)$-time algorithm for paths. We also adapt the approach to compute RAC drawings, i.e. drawings where the angles formed at edge crossings are $\frac{\pi}{2}$.

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

Keywords and phrases Tree drawings, Prescribed edge crossings, Thrackle, Curve complexity, Point-set embeddings, RAC drawings, Topological linear embeddings

Digital Object Identifier 10.4230/LIPIcs.GD.2025.8

Related Version *Full Version*: `https://arxiv.org/abs/2508.18535` [5]

Funding *Giuseppe Di Battista*: Research supported by MUR PRIN Project no. 2022TS4Y3N.
Giuseppe Liotta: Research supported by MUR PON Proj. ARS01 00540 and by MUR PRIN Project no. 2022TS4Y3N.
Maurizio Patrignani: Research supported by MUR PRIN Project no. 2022TS4Y3N.
Antonios Symvonis: Supported by HFRI Grant No: 26320.

Acknowledgements This work started at the Bertinoro Workshop on Graph Drawing BWGD 2024.

1 Introduction

In this paper we study the following problem: Given an n-vertex tree T, a set S of n distinct points in the plane, and a non-negative integer χ, find a simple drawing Γ of T with the following properties:

1. Γ is a *point-set embedding* of T on S, i.e. its vertices are the points of S,


© Giuseppe Di Battista, Giuseppe Liotta, Maurizio Patrignani, Antonios Symvonis, and Ioannis G. Tollis;
licensed under Creative Commons License CC-BY 4.0



2. the edges of T are represented by polylines with constant *curve complexity*, measured as the maximum number of bends per edge, and
3. the edges cross exactly χ times in Γ.

In the following we refer to them as Property 1, Property 2, and Property 3.

The problem locates itself at the intersection of three well studied research subjects in Graph Drawing and Graph Theory, namely the study of point-set embeddings, the study of drawings with limited curve complexity, and the study of drawings with prescribed numbers of edge crossings. Before describing our contribution, we briefly recall each such topic.

Property 1 – For a graph G with n vertices and a set S of n distinct points in the plane, a *point-set embedding* of G on S is a drawing of G where each vertex is mapped to a distinct point of S. Most of the literature on this topic is about planar ($\chi = 0$) point-set embeddings of graphs (see, e.g., [7, 8, 9, 10, 11, 13, 35, 43, 44]). Specifically, it is NP-complete to test if a planar graph G has a straight-line planar point-set embedding on a given point-set [13]. If G is an outerplanar graph and the points are in general position such a drawing can be constructed in polynomial time (see, e.g., [8, 9, 10, 11]).

Property 2 – Together with the edge crossing minimization, the minimization of the edge bends and/or of the bends per edge are among the oldest optimization questions in graph drawing (see, e.g.,[15, 38, 50]). In particular, the relationship between point-set embeddings and curve complexity has been studied when the mapping between the vertices of the graph and the points is specified as part of the input as well as when it is partially specified or not specified (see, e.g., [3, 29, 30, 31, 32, 33, 34, 39, 45, 46]). Notably, Kaufmann and Wiese proved that when the mapping is not specified a planar graph admits a crossing-free point-set embedding with constant curve complexity, namely at most two bends per edge [39, 40].

Property 3 – The *thrackle bound* $\vartheta(G)$ of a graph $G = (V, E)$ with $m = |E|$ edges is the number of crossings that a drawing of G would have if every edge crosses every other non-adjacent edge exactly once. It is well known that $\vartheta(G) = (m(m+1) - \sum_{v \in V} \deg^2(v))/2$; see, e.g., [48, 49]. As an example, for a cycle C_n with n vertices the above formula gives $\vartheta(C_n) = n(n-3)/2$. For a path P_n with n vertices, where all but two vertices have degree two, the formula gives $\vartheta(P_n) = (n-2)(n-3)/2$. A *thrackle* is a drawing of a graph G with exactly $\vartheta(G)$ crossings. Not all graphs admit thrackles. For example, a cycle C_4 with 4 vertices does not admit a drawing with $\vartheta(C_4) = 2$ crossings. As reported in [51], Conway conjectured in 1969 that each thrackleable graph contains at most as many edges as vertices. His conjecture, known as *Conway's thrackle conjecture,* still remains open. Woodall [51] was the first to work on the thrackle conjecture and, assuming the conjecture was true, showed that a finite graph admits a thrackle if and only if it contains at most one odd cycle, no 4-cycle, and each of its connected components is either a tree or it contains exactly one cycle. Refer to [26, 27, 41, 52] and the references therein for progress relevant to Conway's Thrackle conjecture. Graph classes that admit thrackles (and satisfy the thrackle conjecture) include cycles of more than 4 vertices [36] and trees [48]. Given a tree T and an integer $0 \leq \chi \leq \vartheta(T)$, Piazza et al. [48] presented an algorithm that constructs a drawing of T with χ crossings and curve complexity $O(n)$. A similar result was presented by Harborth [36] for any cycle C_n , $n > 4$. Hence, the algorithms in [48] and [36] compute drawings that satisfy Property 3.

Relationship between Properties 2 and 3 – A *linear thrackle* is a thrackle with curve complexity zero. Not all graphs admit linear thrackles. For example, a cycle C_6 with 6 edges does not admit a linear thrackle with $\vartheta(C_6) = 9$ crossings (but it admits a thrackle [48]). It should be noted that, as proved by Erdős [24] and Perles [47], Conway's conjecture holds for linear thrackles. More generally, there is a complex relationship between number of

crossings and curve complexity. For example, a cycle C with an odd number of vertices admits a drawing with curve complexity zero and χ crossings, where χ is any number up to $\vartheta(C)$ except $\vartheta(C)-1$ [28]. On the other hand, an n-vertex cycle C with an even number of vertices admits a drawing with complexity zero and χ crossings, where χ is any number up to $n(n-4)/2+1 = \vartheta(C) - n/2 + 1$ [28]. Notice that there exist planar graphs admitting straight-line pointset embeddings on non-convex pointsets that have more crossings than any straight-line pointset embedding on a pointset in convex position [14]

Besides the above properties we are also interested in producing RAC drawings. A seminal user-study by Huang, Eades, and Hong [37] shows that edge crossings that form sharp angles affect the readability of a drawing much more than those that form large angles. This has motivated the study of right-angle crossing drawings (*RAC drawings*), that are graph drawings where the crossing angles are all $\frac{\pi}{2}$; see, e.g., [1, 2, 6, 16, 18, 19, 17, 20, 21, 22, 23] for surveys and papers about RAC drawings. Point-set RAC embeddings were studied by Fink et al. [25] who, among other results, show that every graph admits a point-set RAC embedding with curve complexity 3.

Our Contribution. We solve the main problem in this paper for any $0 \leq \chi \leq \vartheta(T)$. To this aim, we first present an $O(n)$-time algorithm which *tangles* T by computing a linear layout with $\vartheta(T)$ edge crossings where each edge traverses the spine (i.e. the line passing through the vertices) a constant number of times. We then show how to *untangle* the linear layout until we reach the desired number χ of edge crossings. By carefully *pruning* some edges of T, such a linear layout can be computed in $O(n^2)$ time. If T is a path the time to construct the linear layout reduces to $O(n)$. Finally, the linear layout is used to construct a point-set embedding Γ of T having χ edge crossings and curve complexity at most 5. If we require Γ to be a RAC drawing, then Γ has curve complexity at most 9, which decreases to 6 if the points are collinear. For the case of paths, a RAC point-set embedding with curve complexity 3 can be computed in $O(n \log n)$ time.

Full proofs of statements marked with a "$\star$" can be found in [5].

2 Preliminaries

We only consider *simple* graphs, i.e. graphs not containing loops and multiple edges. A *drawing* of a graph $G(V,E)$ maps each vertex $v \in V$ to a distinct point on the plane and each edge $(u,v) \in E$ to a Jordan arc joining the points representing u and v. A drawing is *simple* if: (i) two adjacent edges do not intersect except at a common end-vertex; (ii) no three edges intersect on the same point; (iii) two edges intersect at most once at a common interior point; and (iv) the intersection between a pair of edges must be "transverse", i.e., the curves of the two edges must alternate around the intersection point. We only consider simple drawings of graphs which we shall call just *drawings*, for short. A drawing is *straight-line* if the edges are represented as straight-line segments. A drawing is *polyline* if each edge is represented as a chain of straight-line segments such that any consecutive segments of the chain have different slopes. The *curve complexity* of a polyline drawing is the maximum number of bends per edge in the drawing. A drawing is a *RAC drawing* if any two edges that cross do so by forming four $\frac{\pi}{2}$ angles around the crossing point.

A *topological linear embedding* $\mathcal{E}$ of graph $G(V,E)$ is defined as follows. (1) Each edge (u,v) of E is mapped to a *subdivision path* π composed of $k>0$ *legs* $\langle (u_0,u_1),(u_1,u_2),\ldots,(u_{k-1},u_k)\rangle$ with $u_0=u$ and $u_k=v$, possibly consisting of a single leg (u,v). The *subdivision vertices* $u_1,u_2,\ldots,u_{k-1}$ internal to π are *spine traversals*. (2) The vertices in V and the

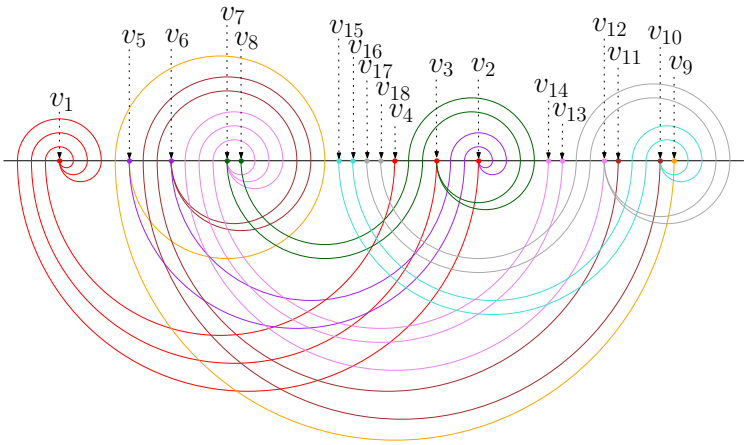


Figure 1 A topological linear layout of the tree of Fig. 2a obtained from the topological circular layout of Fig. 4.

spine traversals are ordered in $\mathcal{E}$; this order is the *spine* of $\mathcal{E}$. (3) The legs are subdivided into two partition sets named *top page* and *bottom page* in such a way that any two consecutive legs of a subdivision path are assigned to distinct partition sets. (4) Two legs (a, b) and (c, d) in the same partition set *cross* if $a \succ c \succ b \succ d$. (4.1) No two legs of the same edge can cross. (4.2) Edges (u, v) and (w, z) cross ρ times if there are ρ distinct pairs of their legs that cross. We impose $\rho \leq 1$. If $\rho = 0$ (u, v) and (w, z) *do not cross in* $\mathcal{E}$, whereas, if $\rho = 1$ they do.

A drawing Γ of a topological linear embedding $\mathcal{E}$ can be constructed as follows. The vertices and the spine traversals of $\mathcal{E}$ are represented in Γ as points along a horizontal line, called *spine* of Γ, in the same left-to-right order they have in $\mathcal{E}$. The half-plane above (resp. below) the spine is the *top page* (resp. *bottom page*) of Γ. A leg (a, b) of G assigned to the top page (resp. bottom page) in $\mathcal{E}$ is represented as the arc of a circumference in the top page (resp. bottom page) of Γ having the points corresponding to a and b along the spine as the extreme points of the diameter. Such arc is a *leg* of Γ. As a consequence, two legs cross in Γ if and only if they cross in $\mathcal{E}$. If a is a spine traversal in the topological linear embedding, the point representing a along the spine of Γ is a *spine traversal* of Γ. Drawing Γ is a *topological linear layout* of G. See Fig. 1. It is immediate to see that, given a topological linear embedding of a graph G, a topological linear layout of G can be constructed in $O(n + \sigma m)$ time, where n, m and σ denote the number of vertices, edges, and maximum number of spine traversals per edge of G, respectively.

3 Topological Linear Embeddings of Trees

In this section we describe Algorithm Tangle, that computes in $O(n)$ time a topological linear embedding of an n-vertex tree T with $\vartheta(T)$ crossings and with two subdivision vertices per edge. Second, we describe Algorithm Untangle which can be iteratively used to reduce the number of crossings from $\vartheta(T)$ down to zero, hence guaranteeing the existence of a topological linear embedding with χ crossings, $\chi \in [1..\vartheta(T)]$ and with at most two subdivisions per edge. Since $\vartheta(T) \in O(n^2)$ and each step of Untangle takes $O(n)$ time, this leads to an $O(n^3)$-time algorithm to produce a topological linear embedding with χ crossings, $\chi \in [1..\vartheta(T)]$. We will also show how to reduce this time to $O(n^2)$.

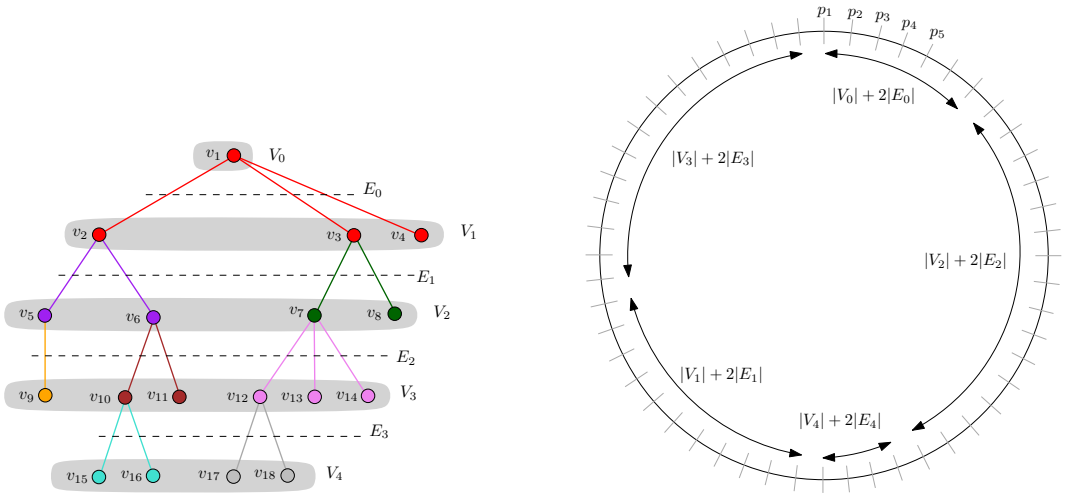


Figure 2 An example tree (a) and the subsequences of points on $\mathcal{C}$ used for its vertices/edges (b).

For the ease of the description, instead of producing a topological linear embedding, we produce a *topological circular embedding* whose definition is analogous to that of a topological linear embedding with the difference that the spine is a circular ordering[1] rather than a linear ordering. A topological linear embedding is obtained from a topological circular embedding and vice versa by considering the spine ordering as linear or circular, respectively. Observe that crossings among the edges and spine traversals are preserved in the two embeddings.

3.1 Algorithm Tangle

In the topological circular embedding $\mathcal{E}_\circ$ of tree $T(V, E)$ constructed by Algorithm Tangle each edge $e \in E$ has two subdivision vertices and, hence, it is split into three legs denoted by $\overset{\circ}{e}$, $\overline{e}$, and $\overline{e}^{\circ}$, where $\overset{\circ}{e}$ is incident to the parent vertex and $\overline{e}^{\circ}$ is incident to the child vertex. We will assign $\overset{\circ}{e}$ and $\overline{e}^{\circ}$ to the inner region (bottom page) and second leg $\overline{e}$ to the outer region (top page). The crossing among legs will happen exclusively in the inner region.

We assume that T is rooted at vertex v_1 and planarly embedded in such a way that for any vertex v_i, with $i = 1, \ldots, n$, the non-leaf children of v_i precede the leaf children of v_i (see, for example, Fig. 2a). Let h be the height of T. For $j = 0, \ldots, h$, we denote by V_j the set of the vertices at depth j in T and by E_j, $j = 0, \ldots, h-1$, the edges between the vertices in V_j and those in V_{j+1}. We choose a sequence $P = \langle p_1, p_2, \ldots, p_{|V|+2|E|} \rangle$ of $|V|+2|E|$ distinct points in this clockwise order along a cycle $\mathcal{C}$. The points in P will be used either by the vertices in V or by the subdivision vertices of the edges in E. We subdivide the points in P into contiguous subsequences of points along $\mathcal{C}$. More specifically, if h is even, we subdivide P into P_0, P_2, P_4, …, P_h, P_1, P_3, P_5, …, P_{h-1}; otherwise, we subdivide P into P_0, P_2, P_4, …, P_{h-1}, P_1, P_3, P_5, …, P_h. Each subsequence P_j ($j = 0, 1, \ldots, h$) has size $|P_j| = |V_j| + 2|E_j|$ and is used by the vertices in V_j and by the subdivision vertices of the edges in E_j (refer to Fig. 2b).

[1] A circular ordering generalizes a linear ordering to settings where there is no "first" or "last" element; we can think it as arranging elements around a circle rather than along a line.

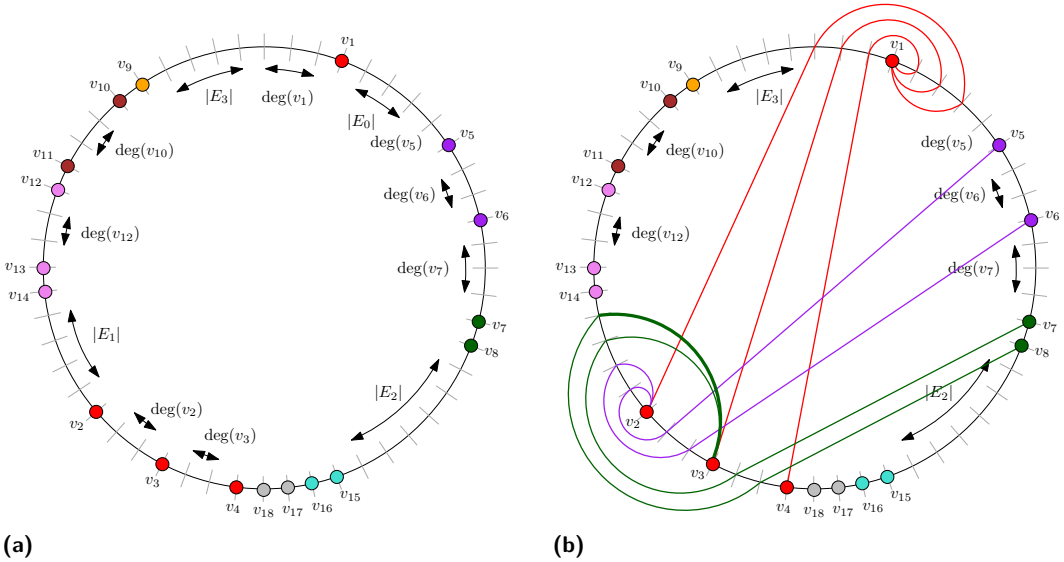


Figure 3 Placement of the vertices (a) and Edge routing (b) of Algorithm Tangle.

Vertex placement. We map the vertices of V_j to specific points of P_j. If j is even (resp. j is odd), denote by $v_{j,1}, v_{j,2}, \dots v_{j,|V_j|}$ the vertices of V_j in the order (resp. in the reversed order) as they appear at depth j in T. For $k = 1, \dots, |V_j|$, we leave children$(v_{j,k})$ points of P_j unused and then we place $v_{j,k}$ (see Fig. 3a). Observe that after the last vertex $v_{j,|V_j|}$ of V_j has been placed, there are $2|E_j|$ unused points, $|E_j|$ of which are at the end of P_j.

Edge routing. We describe how to embed the edges of E_j, for each $j = 0, \dots, h-1$. Denote by $p_{j,1}, p_{j,2}, \dots, p_{j,|V_j|+2|E_j|}$ the points of P_j in this clockwise order around $\mathcal{C}$. Recall that the vertices of V_j have been already placed in P_j and that each vertex $v_{j,k}$, with $k = 1, \dots, |V_j|$, is preceded by children$(v_{j,k})$ unused points, while the whole sequence P_j is closed by $|E_j|$ unused points. For $k = 1, \dots, |V_j|$, we process vertex $v_{j,k}$ of V_j and, for $\ell = 1, \dots,$ children$(v_{j,k})$, we embed each edge $e_\ell \in |E_j|$ incident to $v_{j,k}$, where edges in E_j are considered in the left to right order as they appear in the embedding of T. Intuitively, edge e_ℓ will greedily use the last unused point of P_j and the first unused point of P_j. More formally, we embed $\overset{\circ}{\overline{e_\ell}}$ in the inner region using as its endpoint the last unused point of P_j (refer to Fig. 3b); we embed $\overline{e_\ell}$ in the outer region using as its second endpoint the (children$(v_{j,k}) - \ell + 1$)-th point of P_j clockwise preceding the point used by $v_{j,k}$ (which is also the first unused point of P_j); we embed $\overline{e_\ell}^{\circ}$ in the inner region hitting the point of P_{j+1} where the ℓ-th child of $v_{j,k}$ lies.

▶ **Lemma 1.** *Algorithm Tangle computes in $O(n)$ time a topological circular embedding of an n-vertex tree T with $\vartheta(T)$ crossings where every edge traverses the spine exactly twice.*

Proof sketch. By construction, all the edges of a topological circular embedding $\mathcal{E}_\circ$ produced by Algorithm Tangle are composed by three legs, the second of which is drawn in the outer region. It can be easily shown that the second legs of the edges in E do not cross in $\mathcal{E}_\circ$. Also, the first legs of the edges in E do not cross among themselves in $\mathcal{E}_\circ$. It follows that the only possible crossings in $\mathcal{E}_\circ$ are between a first leg and a third leg of two edges in E or between two third legs of two edges in E. Consider two edges (u, v) and (w, z) of the same E_j, $j \in [0..h-1]$, where u and w belong to P_j. Edges (u, v) and (w, z) cross whenever u uses a point of P_j that precedes the point of P_j used by w ($\overset{\circ}{\overline{(u, v)}}$ crosses $\overline{(w, z)}^{\circ}$). Hence, all the

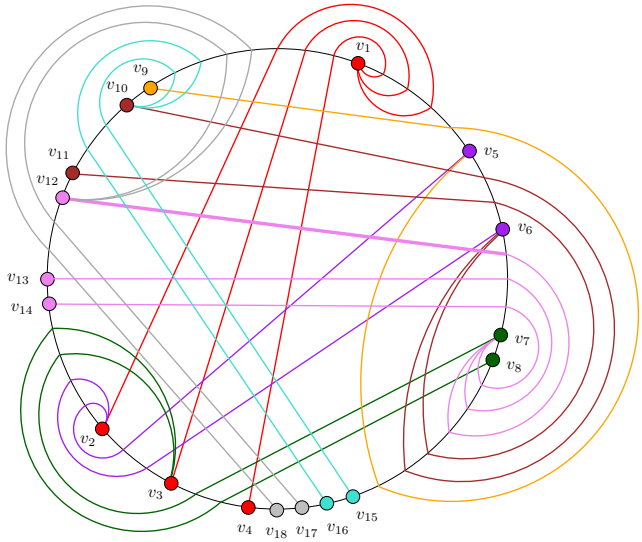


Figure 4 A drawing representing a topological circular embedding of the tree T of Fig. 2a with $\vartheta(T)$ edge crossings and two spine crossings per edge.

edges of E_j cross among themselves with the exception of those that are incident to the same vertex of V_j. Consider two edges $(u, v) \in E_j$ and $(w, z) \in E_{j-1}$, with $j \in [1..h-1]$, where u and z belong to V_j. Edges (u, v) and (w, z) cross when u uses a point of P_j that precedes the point of P_j used by z ($\overset{\circ}{(u, v)}$ crosses $\overline{(w, z)}$). Also, (u, v) and (w, z) cross when u uses a point of P_j that follows the point of P_j used by z ($\overline{(u, v)}$ crosses $\overline{(w, z)}$). Hence, $(u, v) \in E_j$ and $(w, z) \in E_{j-1}$ cross whenever they are not adjacent (i.e., whenever $u \neq z$). Finally, two edges $(u, v) \in E_j$ and $(w, z) \in E_{j'}$ with $j' \notin \{j-1, j, j+1\}$ always cross once ($\overline{(u, v)}$ crosses $\overline{(w, z)}$). In conclusion two edges of T cross if and only if they are not adjacent. Hence the number of crossings is $\vartheta(T)$. It is easy to check that $\mathcal{E}_\circ$ can be computed in $O(n)$ time. ◀

An example drawing of the topological circular embedding computed by Algorithm Tangle is shown in Fig. 4. As a byproduct of the results in this section, we point out that from the circular topological embedding $\mathcal{E}_\circ$ it is not difficult to compute a polyline drawing of T with $\vartheta(T)$ crossings (a thrackle of T) with curve complexity two as in Fig. 5.

3.2 Algorithm Untangle

We describe Algorithm Untangle as if the target were to reduce the number of crossings down to zero. In order to obtain a topological circular embedding $\mathcal{E}_\circ^\chi$ with χ crossings it is sufficient to run $\vartheta(T) - \chi$ iterations of the algorithm.

Let $\vartheta'(T) = \frac{1}{2} \cdot |E|^2 - \frac{1}{2} \cdot \sum_{i=0}^{h-1} |E_i|^2 - \sum_{i=0}^{h-2} (|E_i| \cdot |E_{i+1}|)$. The algorithm has two phases: in the first phase the crossings are reduced from $\vartheta(T)$ to $\vartheta'(T)$, obtaining a topological circular embedding $\mathcal{E}_\circ^{\vartheta'(T)}$ where each edge is subdivided exactly once. The reason for the choice of $\vartheta'(T)$ will become evident soon. In the second phase the number of crossings is reduced to zero.

Phase 1: reducing the crossings to $\vartheta'(T)$. Let $\mathcal{E}_\circ^{\vartheta(T)}$ be a topological circular embedding of T with $\vartheta(T)$ crossings. While reducing at each step the number of crossings by one, we process T level by level bottom-up. In particular, for each $i = h-1, h-2, \ldots, 0$, we first

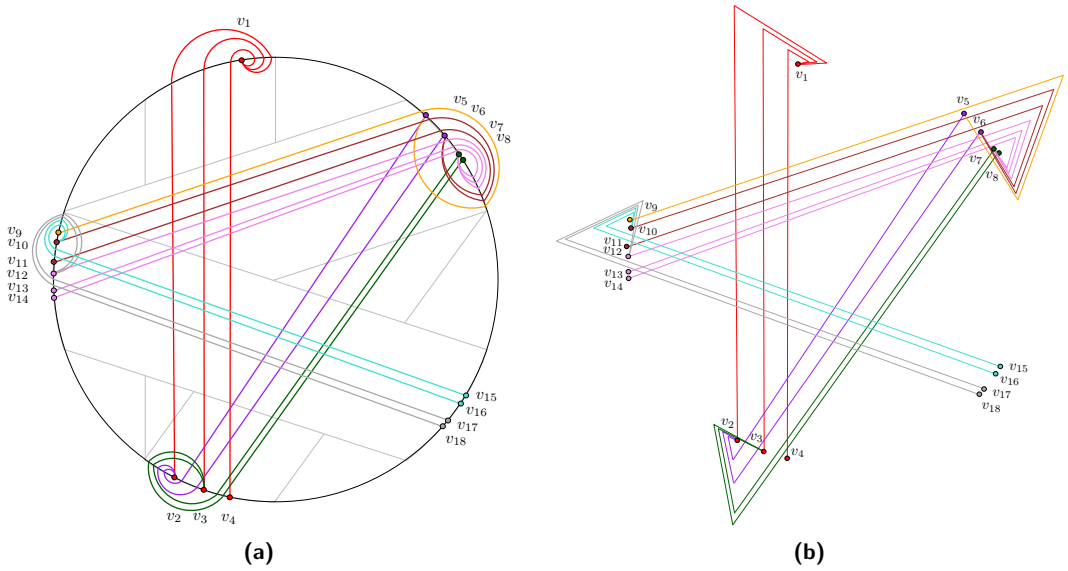


Figure 5 From a topological circular embedding of a tree to a thrackle with curve complexity two.

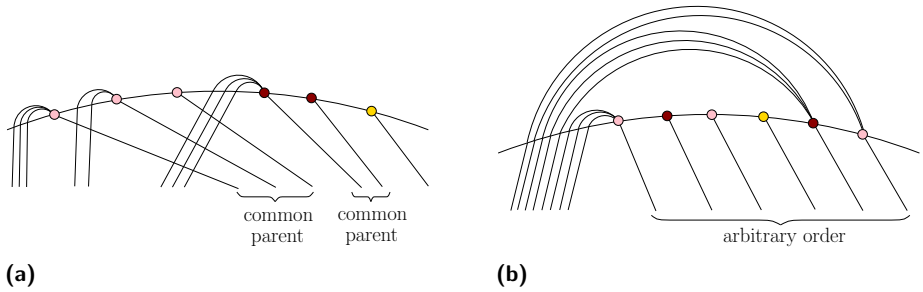


Figure 6 Examples of vertices embedded as rainbows (a) and a full-rainbow configuration (b).

modify the embedding of T in such a way that each vertex of V_i assumes a configuration that we call "rainbow" and we then bring all vertices of V_i and edges of E_i to a configuration that we call "full-rainbow". When all levels of the tree are processed, the number of crossings will be exactly $\vartheta'(T)$ and each edge of T will have exactly one subdivision vertex.

A vertex v is embedded as a *rainbow* if: (i) the edges that lead to its children are subdivided exactly once, where their first leg is assigned to the outer region and their second leg is assigned to the inner region and (ii) the subdivision vertex of such edges immediately precede v in clockwise order. See Fig. 6a for an example of vertices embedded as rainbows. The vertices of V_i are in a *full-rainbow configuration* if: (i) the edges of E_i are planarly drawn with only two legs, the first external and the second internal to $\mathcal{C}$ and (ii) the crossing of an edge (u, v) of E_i with $\mathcal{C}$ clockwise precedes all the vertices of V_i. See Fig. 6b for an example of a full-rainbow configuration. Observe that in a full-rainbow configuration of V_i the first legs of edges of E_i do not cross each other and no edge of E_i crosses an edge of E_{i-1}.

There are as many full-rainbow configurations of V_i as there are permutations of the vertices in V_i. In particular, if we have a full-rainbow configuration and we swap the position of two consecutive vertices v_x and v_y, we can obtain another full-rainbow configuration by swapping the position of the crossings with $\mathcal{C}$ of the edges of E_i incident to v_x with those of E_i incident to v_y, and propagating the swap to the descendants of v_x and v_y. Hence, we obtain the following lemma:

▶ **Lemma 2.** *Let T be a tree of height h. Let $0 \leq i \leq h$ and assume that for all $j \geq i$ the vertices of V_j are in a full-rainbow configuration. There exists a topological circular embedding $\mathcal{E}_\circ$ of T such that the only crossings among edges in $E_i \cup E_{i+1} \cup \dots E_{h-1}$ occur between edges in E_j and E_k with $k > j+1$.*

Note that for $i = 0$ Lemma 2 yields a topological circular embedding $\mathcal{E}_\circ$ of T where the edges of the same E_i do not cross and edges of E_i and E_{i+1} do not cross either. As indicated in Lemma 3, such an embedding has exactly $\vartheta'(T)$ crossings.

Suppose to have a topological circular embedding $\mathcal{E}_\circ^X$ of T such that for a given $0 \leq i \leq h-1$ it holds that: (i) for all $j > i$ the vertices of V_j are in a full-rainbow configuration; (ii) for all $j \leq i$ the vertices of V_j are embedded as in $\mathcal{E}_\circ^{\vartheta(T)}$; and (iii) for all $j < i$ the edges of E_j are embedded as in $\mathcal{E}_\circ^{\vartheta(T)}$. Observe that for $i = h-1$ such a drawing is trivially $\mathcal{E}_\circ^{\vartheta(T)}$. First, we reduce the number of crossings one by one moving all vertices of V_i to a rainbow configuration. Second, we reduce the crossings one by one moving V_i and E_i to a full-rainbow configuration. For the first target, recall that in $\mathcal{E}_\circ^{\vartheta(T)}$ a second leg of an edge in E_i is embedded on the outer region and that the second legs of edges in E_i do not cross. Consider a non-leaf vertex u of V_i and one of its edges $e = (u, v) \in E_i$. If u has more than one child in T, then consider the edge $e \in E_i$ such that the first subdivision vertex x of e is closer to u (refer to Fig. 7). There are three cases: Case 1: If the last point of P_i before x in clockwise order is used by u (see Fig. 7a), then the first subdivision vertex of the edges joining u with its children can be removed, and u can be joined directly to their second subdivision vertices on the outer region without changing the total number of crossings (see Fig. 7b). This yields an embedding of u as a rainbow. Case 2: If the last point of P_i before x in clockwise order is used by a leaf w (see Fig. 7c), then we can swap the position of w and x, decreasing the number of crossings by one, having removed the crossing that was between the edge incident to w and e (see Fig. 7d). This swap operation can be repeated with the other subdivision vertices x', x'', ... of the edges joining u with its children, reducing each time the number of crossings by one (see, for example, Fig. 7e). Case 3: If the last point of P_i before x in clockwise order is used by a non-leaf vertex w, then we can assume that w is embedded as a rainbow (see Fig. 7f). Indeed, if this was not the case we would process w before u. We move w, together with the first subdivision vertices of the edges joining w with its children, clockwise after x (see Fig. 7g). In doing so, we reduce the number of crossings by $\deg(w)$ instead of one unit (compare Figs. 7f and 7g where (u, v) loses four crossings). However, since the vertices of V_{i+1} are in a full-rainbow configuration, they can be permuted and, in particular, the vertex v' that clockwise follows the children of w can be moved counter-clockwise so that the third leg of edge (u, v') crosses children(w) edges of E_i and the total number of crossings is reduced by one unit only (see Fig. 7h where vertex v' coincides with v). Successively, we move v' back to its original position reducing the crossings of one unit at each swap (Fig. 7i shows the first step of such a process). We iterate these operations until none of the above moves apply, i.e., until all the vertices of V_i are in a rainbow configuration.

We now describe how to change the embedding of T in such a way that the vertices in V_i and the edges in E_i are in a full-rainbow configuration. Recall that, since each vertex of V_i has a rainbow configuration, each edge of E_i has a single subdivision vertex. Consider the first subdivision vertex x, in clockwise order, of an edge $e \in E_i$ that is preceded by a vertex $u \in V_i$. Move x before u and before all subdivision vertices of the edges joining u with its children. Observe that this change reduces the number of crossings by one (the crossing between e and the edge of E_{i-1} incident to u). By iterating this process we have that no subdivision vertex of E_i is preceded by a vertex of V_i in P_i, i.e., the vertices of V_i are in a full-rainbow configuration.

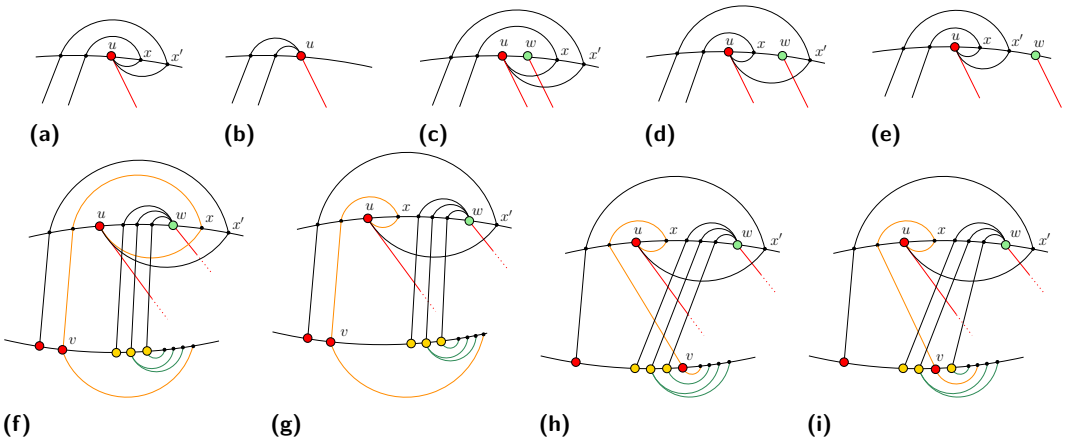


Figure 7 Illustrations for Algorithm Untangle.

▶ **Lemma 3.** *Following the completion of Phase 1, when all V_i, $0 \leq i \leq h$, are in full-rainbow configuration, the total number of crossings is $\vartheta'(T)$.*

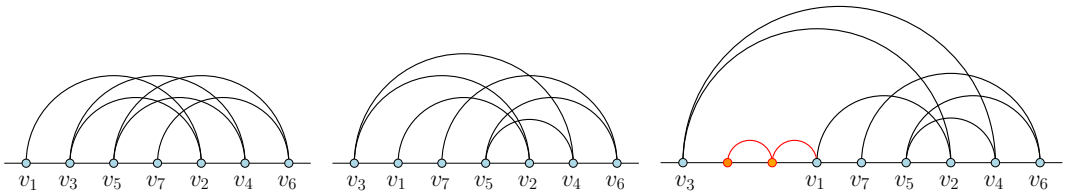
Phase 2: reducing the crossings to zero. When all the levels of the tree are in a full-rainbow configuration, all edges are drawn with a single subdivision vertex. Consider an edge (u, v) incident to a vertex $v \in V_h$ such that v is the last vertex clockwise of V_h. Observe that v is a leaf. Denote by x the subdivision vertex of (u, v). Vertex v can be moved clockwise towards x removing one crossing at each step. When v reaches x, vertex v can be transferred close to its parent u and identified with it (i.e., v can be planarly embedded arbitrarily close to u and moved with u from now on). Iteratively performing this operation on the last vertex of the last level of T reduces the number of crossings down to zero, thus we have:

▶ **Lemma 4.** *Any n-vertex tree T admits a topological linear embedding with χ edge crossings, $\chi \in [0..\vartheta(T)]$, where every edge traverses the spine at most twice. Such an embedding can be computed in $O(n^3)$ time.*

3.3 Algorithm Prune-Tangle-Untangle

In this section we describe an algorithm, called Prune-Tangle-Untangle, that reduces the time complexity of Lemma 4 to $O(n^2)$. The algorithm uses the Tangle and Untangle algorithms. However, before these, it launches the Prune procedure illustrated below.

Let T_n be the input n-vertex tree. The leaves of T_n are recursively removed (in any order) obtaining trees $T_{n-1}, T_{n-2}, \ldots$, while $\vartheta(T_{n-i}) > \chi$. Note that an edge (u, v) of an n-vertex tree T_n, incident to a leaf v, may cross all other $n-2$ edges of T_n except the $\deg(u)-1$ edges incident to u. Hence, removing (u, v) produces a tree T_{n-1} where the number of possible crossings decreases by a quantity in $[0..n]$. To efficiently identify the leaves and compute the current number of crossings, we equip vertices with their degrees and maintain a list of all the leaves. When a leaf is removed from the tree (and from the list) the degree of its parent is decreased, and if it becomes one, the parent is added to the list of leaves. We continue until we encounter a leaf that cannot be removed. Thus, it holds that $\vartheta(T_{n-i}) < \chi + n$. Then, we launch Algorithm Tangle on T_{n-i} to obtain $\mathcal{E}_\circ^{\vartheta(T_{n-i})}$ of T_{n-i} and then perform $\vartheta(T_{n-i}) - \chi$ iterations of Untangle to obtain $\mathcal{E}_\circ^{\chi}$ of T_{n-i}. Finally, the removed edges are planarly added in reverse order as they were removed, iteratively placing them along $\mathcal{C}$ at a small enough distance from their parent, to obtain a topological circular embedding of T with χ crossings. The above description implies the following:



(a) (b) (c)

Figure 8 Topological linear layouts of paths. (a) A linear layout of P_7 with 10 crossings. (b) Linear layout obtained moving vertices v_1 and v_7 in the one in Fig. 8a. $\Delta_\chi = 2$ and $k = 1$. (c) Linear layout computed for path P_9 and $\chi = 8$.

▶ **Theorem 5.** *Any n-vertex tree T admits a topological linear embedding with χ edge crossings, $\chi \in [0..\vartheta(T)]$, where every edge traverses the spine at most twice. Such an embedding can be computed in $O(n^2)$ time.*

4 Topological Linear Embeddings of Paths

In the first part of this section we present an algorithm, which we call χ-*Spiral* that, given an n-vertex ($n \geq 4$) path P_n and any integer χ such that $(n-3)(n-4)/2 < \chi \leq (n-2)(n-3)/2$ computes a spine traversal free, one-page topological linear embedding Γ with χ crossings of P_n. In all topological linear embedding constructed in this section each edge is mapped to just one leg. Hence, we refer to edges and not to legs. Notice also that $(n-3)(n-4)/2 = \vartheta(P_{n-1})$ is exactly the maximum number of crossings for a drawing of a path with $n-1$ vertices.

Algorithm χ-Spiral consists of two steps.

(Step 1) Compute a spine traversal free, one-page topological linear embedding Γ with $\chi' = \vartheta(P_n) = (n-2)(n-3)/2$ crossings of P_n. Let $v_1, \ldots, v_n$ be the vertices of P_n each with a subscript that corresponds to the order in which it appears by visiting P_n from any of its vertices of degree one to the other. Place on the spine of Γ first the vertices with odd subscript, in increasing order of subscript, and then the vertices with even subscript, in increasing order of subscript, too. Hence, the order of the vertices on the spine of Γ is $v_1 \prec v_3 \prec \cdots \prec v_{n-1} \prec v_2 \prec v_4 \prec \cdots \prec v_n$ if n is even and $v_1 \prec v_3 \prec \cdots \prec v_n \prec v_2 \prec v_4 \prec \cdots \prec v_{n-1}$ if n is odd. Then, assign all the edges of P_n to the same page, say top page, of Γ. The linear embedding computed by Step 1 of Algorithm χ-Spiral for path P_7 is shown in Fig. 8a

(Step 2) Let $\Delta_\chi = \chi' - \chi$ be the "excess of crossings" that Γ has with respect to the target χ. Observe that $\Delta_\chi < n-3$. Now, perform what follows. If $\Delta_\chi = 2k+1$ is odd, then move vertex v_1 by $k+1$ positions on the spine to the right, leaving the position of all the other vertices unchanged. Else, ($\Delta_\chi = 2k$ is even) move vertex v_1 on the spine by k positions to the right, leaving the position of all the other vertices unchanged. Then, as a last action of the even case, move vertex v_n to the left by one position. The assignment of all the edges to the same page remains unchanged. The topological linear embedding computed by Algorithm χ-Spiral for path P_7 and $\chi = 8$ (i.e., $\Delta_\chi = 2$) is shown in Fig. 8b.

▶ **Lemma 6.** *Let P_n be an n-vertex ($n \geq 4$) path and let χ be an integer such that $(n-3)(n-4)/2 < \chi \leq (n-2)(n-3)/2$. Algorithm χ-Spiral computes a spine traversal free, one-page topological linear embedding Γ with χ crossings of P_n in $O(n)$ time.*

Proof sketch. First, we prove that at the end of Step 1 Algorithm χ-Spiral gets a topological linear embedding Γ with $\chi' = (n-2)(n-3)/2$ crossings. Since all the edges of P_n are assigned to the same page, Γ is one-page and spine traversal free. We prove that, in Γ, each edge crosses all the other edges but those incident to its end-vertices. Consider any two edges (v_i, v_{i+1}) and (v_j, v_{j+1}) of P_m such that i, j, $i+1$, and $j+1$ are pairwise different and assume, w.l.o.g., that $i < j$. We prove that (v_i, v_{i+1}) and (v_j, v_{j+1}) cross in Γ. Suppose that i and j are both odd, the other cases being similar. We have that, since $i < j$, $v_i \prec v_j$. Also, both $i+1$ and $j+1$ are even, with $i+1 < j+1$, hence, $v_{i+1} \prec v_{j+1}$. Further, since j is odd and $i+1$ is even, v_j is placed before v_{i+1} on the spine ($v_j \prec v_{i+1}$). In summary, $v_i \prec v_j \prec v_{i+1} \prec v_{j+1}$. It follows that (v_i, v_{i+1}) and (v_j, v_{j+1}) cross. The time complexity of Step 1 of the algorithm follows from the construction, which is just a scan of the path.

Concerning Step 2 of the algorithm, we have what follows. (1) Because of the range of χ, we have that $\Delta_\chi < n-3$. (2) If Δ_χ is odd we have $0 \leq k < (n-4)/2$. If Δ_χ is even we have $0 \leq k < (n-3)/2$. Because of these upper bounds on k, after the moves, v_1 is still to the left of v_n in Γ. (3) If v_1 moves by one position to the right, then it is placed right after v_3, hence, the number of crossings in Γ decreases by one (namely (v_1, v_2) no longer crosses (v_3, v_4)). (4) If v_1 moves by $k+1$ positions to the right then the number of crossings in Γ decreases by $1+2k$; e.g. if $k = 1$ then v_1 is positioned between v_5 and v_7, hence, (v_1, v_2) no longer crosses (v_3, v_4), (v_4, v_5), and (v_5, v_6). Essentially, the first move of v_1 to the right decreases the number of crossings in Γ by one, while each of the subsequent moves decreases the number of crossings by exactly two. Observe also that a move of v_n to the left of one position decreases the number of crossings in Γ by one unit. Since Step 2 involves only a linear number of moves on the spine of Γ, its time complexity is also linear. ◀

Now, we exploit Algorithm χ-spiral and Lemma 6 to efficiently produce a drawing of P_n with "any" number of crossings, proving the following lemma.

▶ **Lemma 7.** *Let P_n be an n-vertex path. For each $\chi \in [0..\vartheta(P_n)]$, P_n admits a one-page topological linear embedding with χ crossings, that can be computed in $O(n)$ time.*

Proof. In order to compute the desired linear embedding, we first look for an integer n' such that $\vartheta(P_{n'-1}) < \chi \leq \vartheta(P_{n'})$. Namely, we look for the shortest path $P_{n'}$ such that Lemma 6 can be used to obtain a linear embedding of $P_{n'}$ with the desired number of crossings. This is given by $n' = \lceil \frac{5+\sqrt{25-4(6-2\chi)}}{2} \rceil$.

We then execute Algorithm χ-Spiral to construct a one-page linear embedding Γ' with χ crossings of path $P_{n'}$. Once the one-page linear embedding Γ' of $P_{n'}$ has been computed, we augment Γ' by adding to it a path $P_{n''}$ with $n'' = n - n'$ vertices on the same page of $P_{n'}$ just besides vertex v_1, positioning its vertices consecutively say before v_1 and joining its last vertex with v_1. Let Γ be the obtained linear embedding. We have that the edges of $P_{n''}$ do not cross any edge, hence the total number of crossings of Γ is the same of Γ'. As an example, the linear layout computed for path P_9 and $\chi = 8$ is shown in Fig. 8c. In this case $n' = 7$.

The $O(n)$ time bound easily descends from the description of the algorithm. ◀

5 Point-set Embeddings with Prescribed Edge Crossings

In this section we address the general problem of computing a point-set embedding of a graph G with prescribed edge crossings and $O(1)$ curve complexity. The following lemma extends and refines ideas described in the literature for crossing-free point-set embeddings, [25, 39].

▶ **Lemma 8.** *Let G be a graph with n vertices and m edges. Let S be a set of n distinct points in the plane. If G admits a topological linear embedding with χ edge crossings and at most σ spine traversals per edge, then the following hold:*

- *G admits a RAC point-set embedding on S with χ edge crossings and at most $3(\sigma+1)$ curve complexity, which can be computed in $O(m\sigma + n\log n)$ time.*
- *G admits a point-set embedding on S with χ edge crossings and at most $2\sigma+1$ curve complexity, which can be computed in $O(m\sigma + n\log n)$ time.*

Proof sketch. We assume, w.l.o.g., that no two points of S have the same x-coordinate (this is in fact always the case by suitably rotating the plane). Let L_{top} be a horizontal line above the point of S with the largest y-coordinate and L_{bottom} be a horizontal line below the point of S with the smallest y-coordinate. Let $\mathcal{S}$ be the strip of plane between L_{top} and L_{bottom}. Project the points of S onto the x-axis and order them by increasing x-coordinate. Let Γ be a topological linear layout of G whose spine is the x-axis and whose vertices are the projection points of S onto the spine. For each spine crossing of Γ add a dummy point to S that has the same x-coordinate of the spine crossing and is in the interior of $\mathcal{S}$. Let S' be the set of points that includes all points of S and all dummy points corresponding to the spine crossings of the edges of Γ. In this sketch we show how to construct a RAC point-set embedding, see the appendix for the complete proof.

Since no two points of S' have the same x-coordinate, we can define around each point $s \in S'$ a vertical strip τ_s whose interior contains s and such that no other element of S' is in the interior or on the boundary of τ_s; we call τ_s the *safe strip of* s.

Let u be a vertex of Γ and let $s \in S'$ be the point of S' having the same x-coordinate as u. Let $deg_{top}(u)$ be the number of edges of Γ incident to u in the top page and let $deg_{bottom}(u)$ be the number of edges incident to v in the bottom page. We place $deg_{bottom}(u)$ dummy points in the interior of $\tau_s \cap L_{bottom}$ and $deg_{top}(u)$ dummy points in the interior of $\tau_s \cap L_{top}$. We then connect s to each such dummy points. Each segment connecting s to a dummy point along $\tau_s \cap L_{bottom}$ is a *bottom stub* of s. Each segment connecting s to a dummy point along $\tau_s \cap L_{top}$ is a *top stub* of s. Let a be a spine crossing along an edge of Γ and let $s \in S'$ be the point of S' having the same x-coordinate as a. The *top stub* of s is the vertical segment from s to the projection of s onto L_{top}. The *bottom stub* of s is the vertical segment from s to the projection of s onto L_{bottom}. The projection point of s to L_{top} (L_{bottom}) is referred to as the *endpoint of the top (bottom) stub* of s.

We are now ready to compute a RAC point-set embedding of G. Every vertex or spine crossing of Γ is mapped to the point of S' with the same x-coordinate. For every vertex $u \in \Gamma$ we order its incident edges in the bottom page from left to right; similarly we order from left to right its incident edges in the top page. Let $s \in S'$ be the point onto which we map u: We order both its bottom stubs and its top stubs from left to right. Let (u, v) be the i-th edge of Γ incident to u (either in the bottom or top page of Γ) and the j-th edge incident to v (either in the bottom or top page of Γ).

Let (a, b) be a leg of (u, v) in Γ. Let $s \in S'$ be the point to which a is mapped and let $s' \in S'$ be the point to which b is mapped. Without loss of generality, we assume that $a \prec b$ (the case where $a \succ b$ is handled similarly). If a and b are both spine crossings and (a, b) is in the top (bottom) page of Γ, we connect the endpoint p of the top (bottom) stub of s to the endpoint p' of the top (bottom) stub of s' with a polyline consisting of two segments: the segment incident to p has slope $+1$ (-1), the segment incident to p' has slope -1 ($+1$). Note that the leg (a, b) is mapped to a polyline with three bends: One at p, another at p' and a third one where the two segments with opposite slopes meet. The case where either $a = u$ and/or $b = v$ is treated similarly; the only difference being that the polyline representing

the leg (a, b) in the point-set embedding must be incident to either the i-th stub of the point representing u or to the j-th stub of the point representing v. In this case (a, b) is also represented by a polyline having 3 bends. After every leg of Γ has been drawn in the point-set embedding by means of the above procedure, all dummy points corresponding to the spine crossings of Γ are removed from S'. By construction, every edge of Γ that crosses the spine k times (and consists of $k + 1$ legs) is represented in the point-set embedding as a polyline having $3(k + 1)$ bends. Therefore, if Γ has at most σ spine crossings per edge, the curve complexity of the point-set embedding is $3(\sigma + 1)$.

By construction, the stubs are in the interior of the safe strips and the safe strips do not overlap with each other. This implies that the point-set embedding has no edge crossing in the interior of $\mathcal{S}$. Also, note that the safe regions follow the left to right order of the points around which they are defined: This order is consistent with the left to right order of the vertices and spine crossings along the spine of Γ. It follows that a crossing occurs in the point-set embedding if and only if we have four points s_0, s_1, s_2, s_3 in S' with $x(s_0) < x(s_1) < x(s_2) < x(s_3)$ which correspond to four vertices or spine crossings u_0, u_1, u_2, u_3, respectively, such that in Γ we have an arc of circumference with diameter u_0u_2, another arc of circumference with diameter u_1u_3, such that both legs are in the same page and and $u_0 \prec u_1 \prec u_2 \prec u_3$ in the linear order. This implies that the point-set embedding of G onto S has the same number χ of edge crossings as the topological linear embedding of G.

Since all segments along the polylines outside $\mathcal{S}$ have either slope $+1$ or slope -1 and since any crossing occurs outside $\mathcal{S}$, we also have that the edge crossings form $\frac{\pi}{2}$ angles and hence the point-set embedding is a RAC drawing.

Concerning the time complexity, the procedure in this sketch first sorts the points of S by increasing x-coordinate, then it maps the vertices of G to the points of S and then it processes an edge at a time. It spends constant time to draw any leg. It follows that all edges of the point-set embedding are processed in $O(m\sigma)$-time, which leads to a computational complexity of $O(m\sigma + n \log n)$ to compute a (RAC) point-set embedding of G. ◀

Thm. 5, Lemma 7, and Lemma 8 imply that any tree (resp. path) admits a RAC point-set embedding with a prescribed number of crossings and curve complexity 9 (resp. 3). It can be proved that if the points are collinear, the curve complexity of the RAC point-set embedding can be reduced to 6 (see Fig. 9 for an example and the appendix for details). If we drop the constraint that the edge crossings form right angles, we obtain smaller curve complexities by means of the same Lemmas, as it is summarized in the following theorem.

▶ **Theorem 9.** *Let T be a tree with n vertices, let $\chi \in [0..\vartheta(T)]$, and let S be a set of n distinct points in the plane. The following hold:*

- *We can compute in $O(n^2)$-time a point-set embedding of T on S with χ edge crossings and curve complexity 9 if the drawing is RAC and 5 otherwise. Also, if S is a set of collinear points, the curve complexity of the RAC point-set embedding reduces to 6.*
- *If T is a path, we can compute in $O(n \log n)$-time a point-set embedding of T on S with χ edge crossings and curve complexity 3 if the drawing is RAC and 1 otherwise.*

6 Final Remarks and Open Problems

Graph Drawing is characterized by a range of computational aesthetic metrics [12, 53]. To investigate the impact of each metric on the readability it is useful to produce different drawings of the same graph where all metrics but one remain fixed [42]. In this context, we compute drawings of trees with any number of crossings (which is probably the most studied

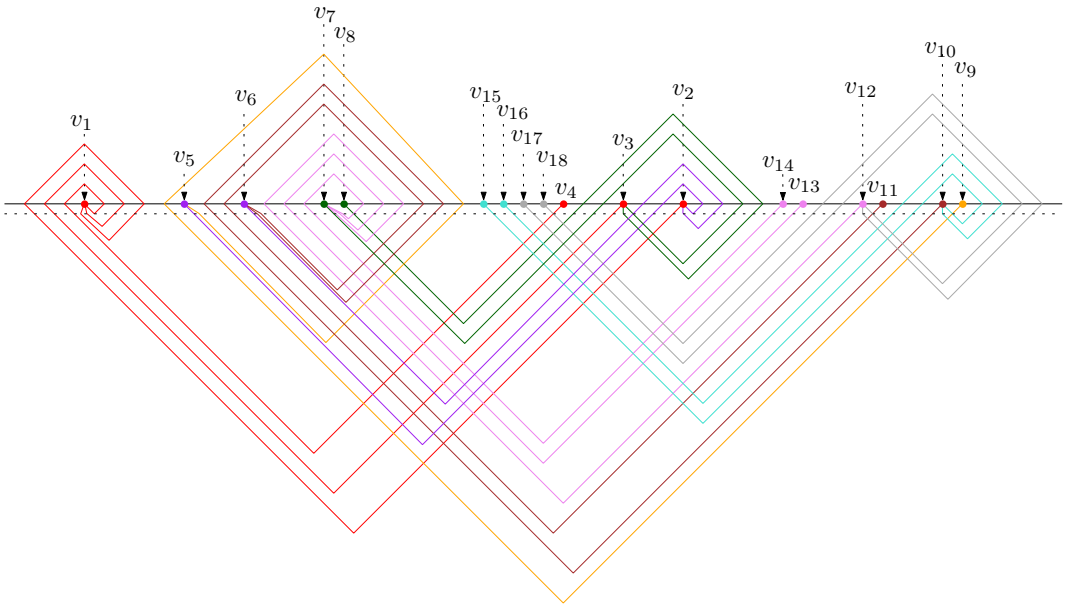


Figure 9 A RAC point-set embedding on collinear points of the tree of Fig. 2a.

aesthetic), while guaranteeing constant curve complexity, and constraining the vertices at given locations.

We conclude with several open problems.

1. Is there an $o(n^2)$-time algorithm to compute a point-set embedding of a tree with χ crossings and constant curve complexity? This question is interesting even for binary trees.
2. Is it possible to compute RAC point-set embeddings of trees with any number of edge crossings and curve complexity smaller than 9?
3. A seminal paper by Pach and Wenger studies the point-set embeddability without crossings when the mapping between vertices and points is part of the input [45, 46]. It would be interesting to study the question of the present paper also in such a setting.
4. Further, it would be interesting to extend our investigation to graph classes other than trees.

Finally, we believe that examining well-known theoretical problems from an unconventional perspective (see, e.g., [4]) can yield new and interesting insights.

References

1 Patrizio Angelini, Luca Cittadini, Giuseppe Di Battista, Walter Didimo, Fabrizio Frati, Michael Kaufmann, and Antonios Symvonis. On the perspectives opened by right angle crossing drawings. In David Eppstein and Emden R. Gansner, editors, *Graph Drawing, 17th International Symposium, GD 2009, Chicago, IL, USA, September 22-25, 2009. Revised Papers*, volume 5849 of *Lecture Notes in Computer Science*, pages 21–32. Springer, 2009. doi:10.1007/978-3-642-11805-0_5.

2 Patrizio Angelini, Luca Cittadini, Walter Didimo, Fabrizio Frati, Giuseppe Di Battista, Michael Kaufmann, and Antonios Symvonis. On the perspectives opened by right angle crossing drawings. *J. Graph Algorithms Appl.*, 15(1):53–78, 2011. doi:10.7155/JGAA.00217.

3 Melanie Badent, Emilio Di Giacomo, and Giuseppe Liotta. Drawing colored graphs on colored points. *Theor. Comput. Sci.*, 408(2-3):129–142, 2008. doi:10.1016/J.TCS.2008.08.004.

4 Giuseppe Di Battista, Fabrizio Frati, and Maurizio Patrignani. On embedding a graph in the grid with the maximum number of bends and other bad features. *Theory Comput. Syst.*, 44(2):143–159, 2009. doi:10.1007/S00224-008-9115-0.

5 Giuseppe Di Battista, Giuseppe Liotta, Maurizio Patrignani, Antonios Symvonis, and Ioannis G. Tollis. Tangling and untangling trees on point-sets. Tech. Report arXiv:2508.18535, Cornell University, 2025. doi:10.48550/arXiv.2508.18535.

6 Michael A. Bekos, Walter Didimo, Giuseppe Liotta, Saeed Mehrabi, and Fabrizio Montecchiani. On RAC drawings of 1-planar graphs. *Theor. Comput. Sci.*, 689:48–57, 2017. doi:10.1016/J.TCS.2017.05.039.

7 Therese Biedl and Martin Vatshelle. The point-set embeddability problem for plane graphs. In *Proceedings of the twenty-eighth annual symposium on Computational geometry*, pages 41–50, 2012. doi:10.1145/2261250.2261257.

8 Prosenjit Bose. On embedding an outer-planar graph in a point set. In Giuseppe Di Battista, editor, *Graph Drawing, 5th International Symposium, GD '97, Rome, Italy, September 18-20, 1997, Proceedings*, volume 1353 of *Lecture Notes in Computer Science*, pages 25–36. Springer, 1997. doi:10.1007/3-540-63938-1_47.

9 Prosenjit Bose. On embedding an outer-planar graph in a point set. *Comput. Geom.*, 23(3):303–312, 2002. doi:10.1016/S0925-7721(01)00069-4.

10 Prosenjit Bose, Michael McAllister, and Jack Snoeyink. Optimal algorithms to embed trees in a point set. In Franz-Josef Brandenburg, editor, *Graph Drawing, GD '95*, volume 1027 of *LNCS*, pages 64–75. Springer, 1995. doi:10.1007/BFB0021791.

11 Prosenjit Bose, Michael McAllister, and Jack Snoeyink. Optimal algorithms to embed trees in a point set. *J. Graph Algorithms Appl.*, 1(2):1–15, 1997. doi:10.7155/JGAA.00002.

12 Michael Burch, Weidong Huang, Mathew Wakefield, Helen C. Purchase, Daniel Weiskopf, and Jie Hua. The state of the art in empirical user evaluation of graph visualizations. *IEEE Access*, 9:4173–4198, 2021. doi:10.1109/ACCESS.2020.3047616.

13 Sergio Cabello. Planar embeddability of the vertices of a graph using a fixed point set is NP-hard. *Journal of Graph Algorithms and Applications*, 10(2):353–363, 2006. doi:10.7155/JGAA.00132.

14 Markus Chimani, Stefan Felsner, Stephen Kobourov, Torsten Ueckerdt, Pavel Valtr, and Alexander Wolff. On the maximum crossing number. *Journal of Graph Algorithms and Applications*, 22(1):67–87, March 2018. doi:10.7155/jgaa.00458.

15 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.

16 Emilio Di Giacomo, Walter Didimo, Peter Eades, and Giuseppe Liotta. 2-layer right angle crossing drawings. *Algorithmica*, 68(4):954–997, 2014. doi:10.1007/S00453-012-9706-7.

17 Walter Didimo. Right angle crossing drawings of graphs. In Seok-Hee Hong and Takeshi Tokuyama, editors, *Beyond Planar Graphs, Communications of NII Shonan Meetings*, pages 149–169. Springer, 2020. doi:10.1007/978-981-15-6533-5_9.

18 Walter Didimo, Peter Eades, and Giuseppe Liotta. A characterization of complete bipartite RAC graphs. *Inf. Process. Lett.*, 110(16):687–691, 2010. doi:10.1016/J.IPL.2010.05.023.

19 Walter Didimo, Peter Eades, and Giuseppe Liotta. Drawing graphs with right angle crossings. *Theor. Comput. Sci.*, 412(39):5156–5166, 2011. doi:10.1016/J.TCS.2011.05.025.

20 Walter Didimo, Giuseppe Liotta, Saeed Mehrabi, and Fabrizio Montecchiani. 1-bend RAC drawings of 1-planar graphs. In Yifan Hu and Martin Nöllenburg, editors, *Graph Drawing and Network Visualization - GD 2016*, volume 9801 of *LNCS*, pages 335–343. Springer, 2016. doi:10.1007/978-3-319-50106-2_26.

21 Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani. A survey on graph drawing beyond planarity. *ACM Comput. Surv.*, 52(1), February 2019. doi:10.1145/3301281.

22 Peter Eades and Giuseppe Liotta. Right angle crossing graphs and 1-planarity. In Marc J. van Kreveld and Bettina Speckmann, editors, *Graph Drawing - GD 2011*, volume 7034 of *LNCS*, pages 148–153. Springer, 2011. doi:10.1007/978-3-642-25878-7_15.

23 Peter Eades and Giuseppe Liotta. Right angle crossing graphs and 1-planarity. *Discret. Appl. Math.*, 161(7-8):961–969, 2013. doi:10.1016/J.DAM.2012.11.019.

24 P. Erdös. On sets of distances of n points. *The American Mathematical Monthly*, 53(5):248–250, 1946. doi:10.2307/2305092.

25 Martin Fink, Jan-Henrik Haunert, Tamara Mchedlidze, Joachim Spoerhase, and Alexander Wolff. Drawing graphs with vertices at specified positions and crossings at large angles. In Md. Saidur Rahman and Shin-Ichi Nakano, editors, *WALCOM: Algorithms and Computation - WALCOM 2012*, volume 7157 of *LNCS*, pages 186–197. Springer, 2012. doi:10.1007/978-3-642-28076-4_19.

26 Radoslav Fulek and János Pach. Thrackles: An improved upper bound. In Fabrizio Frati and Kwan-Liu Ma, editors, *Graph Drawing and Network Visualization - GD 2017*, volume 10692 of *LNCS*, pages 160–166. Springer, 2017. doi:10.1007/978-3-319-73915-1_14.

27 Radoslav Fulek and János Pach. Thrackles: An improved upper bound. *Discrete Applied Mathematics*, 259:226–231, 2019. doi:10.1016/j.dam.2018.12.025.

28 W. H. Furry and D. J. Kleitman. Maximal rectilinear crossing of cycles. *Studies in Applied Mathematics*, 56(2):159–167, 1977. doi:10.1002/sapm1977562159.

29 Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, Henk Meijer, Francesco Trotta, and Stephen K. Wismath. k-colored point-set embeddability of outerplanar graphs. In Michael Kaufmann and Dorothea Wagner, editors, *Graph Drawing, GD 2006*, volume 4372 of *LNCS*, pages 318–329. Springer, 2006. doi:10.1007/978-3-540-70904-6_31.

30 Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, Henk Meijer, Francesco Trotta, and Stephen K. Wismath. k-colored point-set embeddability of outerplanar graphs. *J. Graph Algorithms Appl.*, 12(1):29–49, 2008. doi:10.7155/JGAA.00158.

31 Emilio Di Giacomo, Leszek Gasieniec, Giuseppe Liotta, and Alfredo Navarra. Colored point-set embeddings of acyclic graphs. In Fabrizio Frati and Kwan-Liu Ma, editors, *Graph Drawing and Network Visualization - GD 2017*, volume 10692 of *LNCS*, pages 413–425. Springer, 2017. doi:10.1007/978-3-319-73915-1_32.

32 Emilio Di Giacomo, Leszek Gasieniec, Giuseppe Liotta, and Alfredo Navarra. On the curve complexity of 3-colored point-set embeddings. *Theor. Comput. Sci.*, 846:114–140, 2020. doi:10.1016/J.TCS.2020.09.027.

33 Emilio Di Giacomo, Giuseppe Liotta, and Francesco Trotta. Drawing colored graphs with constrained vertex positions and few bends per edge. In Seok-Hee Hong, Takao Nishizeki, and Wu Quan, editors, *Graph Drawing, GD 2007*, volume 4875 of *LNCS*, pages 315–326. Springer, 2007. doi:10.1007/978-3-540-77537-9_31.

34 Emilio Di Giacomo, Giuseppe Liotta, and Francesco Trotta. Drawing colored graphs with constrained vertex positions and few bends per edge. *Algorithmica*, 57(4):796–818, 2010. doi:10.1007/S00453-008-9255-2.

35 Peter Gritzmann, Bojan Mohar, János Pach, and Richard Pollack. Embedding a planar triangulation with vertices at specified points. *The American Mathematical Monthly*, 98(2):165, 1991.

36 Heiko Harborth. Drawings of the cycle graph. *Congressus Numerantium*, 66:15–22, January 1988.

37 Weidong Huang, Peter Eades, and Seok-Hee Hong. Larger crossing angles make graphs easier to read. *J. Vis. Lang. Comput.*, 25(4):452–465, 2014. doi:10.1016/J.JVLC.2014.03.001.

38 Michael Kaufmann and Dorothea Wagner, editors. *Drawing Graphs, Methods and Models (the book grow out of a Dagstuhl Seminar, April 1999)*, volume 2025 of *Lecture Notes in Computer Science*. Springer, 2001. doi:10.1007/3-540-44969-8.

39 Michael Kaufmann and Roland Wiese. Embedding vertices at points: Few bends suffice for planar graphs. *J. Graph Algorithms Appl.*, 6(1):115–129, 2002. doi:10.7155/JGAA.00046.

40 Michael Kaufmann and Roland Wiese. Maintaining the mental map for circular drawings. In Stephen G. Kobourov and Michael T. Goodrich, editors, *Graph Drawing, GD 2002*, volume 2528 of *LNCS*, pages 12–22. Springer, 2002. doi:10.1007/3-540-36151-0_2.

41 L. Lovász, J. Pach, and M. Szegedy. On Conway's thrackle conjecture. *Discrete and Computational Geometry*, 18(4):369–376, 1997. doi:10.1007/PL00009322.

42 Gavin J. Mooney, Helen C. Purchase, Michael Wybrow, and Stephen G. Kobourov. The multi-dimensional landscape of graph drawing metrics. In Jorji Nonaka, editor, *Proceedings - 2024 IEEE 17th Pacific Visualization Conference, PacificVis 2024*, pages 122–131, 2024. doi:10.1109/PacificVis60374.2024.00022.

43 Rahnuma Islam Nishat, Debajyoti Mondal, and Md. Saidur Rahman. Point-set embeddings of plane 3-trees - (extended abstract). In Ulrik Brandes and Sabine Cornelsen, editors, *Graph Drawing - GD 2010*, volume 6502 of *LNCS*, pages 317–328. Springer, 2010. doi:10.1007/978-3-642-18469-7_29.

44 Rahnuma Islam Nishat, Debajyoti Mondal, and Md Saidur Rahman. Point-set embeddings of plane 3-trees. *Computational Geometry*, 45(3):88–98, 2012. doi:10.1016/J.COMGEO.2011.09.002.

45 János Pach and Rephael Wenger. Embedding planar graphs at fixed vertex locations. In Sue Whitesides, editor, *Graph Drawing, GD'98*, volume 1547 of *LNCS*, pages 263–274. Springer, 1998. doi:10.1007/3-540-37623-2_20.

46 János Pach and Rephael Wenger. Embedding planar graphs at fixed vertex locations. *Graphs Comb.*, 17(4):717–728, 2001. doi:10.1007/PL00007258.

47 János Pach and Ethan Sterling. Conway's conjecture for monotone thrackles. *The American Mathematical Monthly*, 118(6):544–548, 2011. doi:10.4169/amer.math.monthly.118.06.544.

48 B. L. Piazza, R. D. Ringeisen, and S. K. Stueckle. Properties of non-minimum crossings for some classes of graphs. In Yousef Alavi, G. Chartrand, O. R. Oellermann, and A. J. Schwenk, editors, *Graph Theory, Combinatorics, and Applications: Proceedings of the Sixth Quadrennial International Conference on the Theory and Applications of Graphs, Kalamazoo, Western Michigan University, 1988, Volume 2*, pages 975–989. Wiley-Interscience, New York, March 1991.

49 Marcus Schaefer. The graph crossing number and its variants: A survey. *The Electronic Journal of Combinatorics*, 2013. Version May 17, 2024. doi:10.37236/2713.

50 Roberto Tamassia, editor. *Handbook on Graph Drawing and Visualization*. Chapman and Hall/CRC, 2013. URL: https://www.crcpress.com/Handbook-of-Graph-Drawing-and-Visualization/Tamassia/9781584884125.

51 D.R. Woodall. Thrackles and deadlock. In D. J. A. Welsh, editor, *Combinatorial Mathematics and its Applications, Proc. 1969 Oxford Combinatorial Conference*, pages 335–347. Academic Press, 2011.

52 Yian Xu. A new upper bound for Conway's thrackles. *Applied Mathematics and Computation*, 389:125573, 2021. doi:10.1016/j.amc.2020.125573.

53 Vahan Yoghourdjian, Daniel Archambault, Stephan Diehl, Tim Dwyer, Karsten Klein, Helen C. Purchase, and Hsiang-Yun Wu. Exploring the limits of complexity: A survey of empirical studies on graph visualisation. *Vis. Informatics*, 2(4):264–282, 2018. doi:10.1016/J.VISINF.2018.12.006.

The Page Number of Monotone Directed Acyclic Outerplanar Graphs Is Four or Five

Jawaherul Md. Alam
Amazon Inc., Tempe, AZ, USA

Michael A. Bekos
Department of Mathematics, University of Ioannina, Greece

Martin Gronemann
Independent Researcher, Dortmund, Germany

Michael Kaufmann
Institut für Informatik, Universität Tübingen, Germany

Abstract

A *k-page book embedding* of a directed acyclic graph consists of a topological order of its vertices and a k-coloring of its edges, such that no two edges of the same color cross, that is, their endpoints do not alternate in the order. The minimum value of k for which such an embedding exists is referred to as the *page number* of the graph. In contrast to general directed acyclic planar graphs, which may have unbounded page number [SIAM J. Comput. 28(5), 1999], it was recently shown that directed acyclic outerplanar graphs have bounded page number. In particular, Jungeblut, Merker and Ueckerdt provided an upper bound of 24,776 on their page number [FOCS 2023: 1937-1952].

In this work, we focus on so-called *monotone* directed acyclic outerplanar graphs. Starting from a single edge, these graphs are constructed by iteratively connecting a new vertex to the endpoints of an existing edge on the outer face using either two incoming or two outgoing edges incident to it. These graphs have *twist-number* 4 [GD 2023: 135-151] (i.e., they admit a topological order in which no more than four edges pairwise cross), a property, which was leveraged by Jungeblut, Merker and Ueckerdt to show that their page number is at most 128. We lower this upper bound to 5 and we also provide a lower bound of 4. A notable consequence of our result is a significant improvement of the upper bound on the page number of general directed outerplanar graphs from 24,776 to 1,160.

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

Keywords and phrases Book embeddings, page number, directed outerplanar graphs

Digital Object Identifier 10.4230/LIPIcs.GD.2025.9

Funding The work of M.B. has been supported by HFRI Grant No: 26320.

Acknowledgements We would like to thank the anonymous reviewers of this submission for valuable comments; in particular, for bringing to our attention the details of Remark 5.

1 Introduction

Embedding graphs in books forms a central topic in topological Graph Theory and Graph Drawing with early results dating back to the 70's [4, 19]. Primarily motivated by applications in VLSI design [7, 20, 21], book embeddings of graphs have been the subject of intensive research [1, 2, 6, 11, 12, 22, 23]. Formally, a *k-page book embedding* of a graph consists of a linear order of its vertices and a k-coloring of its edges, such that no two edges of the same color cross, that is, their endpoints do not alternate in the order; refer, e.g., to [10] for a thorough introduction. When the graph is directed, the underline linear order must coincide with a topological order of the graph [18]; see, e.g., Figure 1. Given a graph (directed or not),


© Jawaherul Md. Alam, Michael A. Bekos, Martin Gronemann, and Michael Kaufmann;
licensed under Creative Commons License CC-BY 4.0



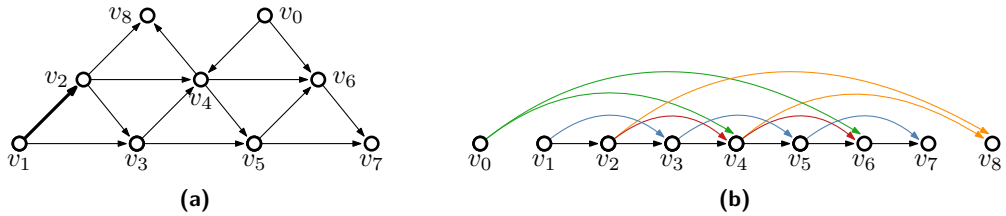


Figure 1 (a) A directed acyclic monotone outerplanar graph with base edge (v_1, v_2), and (b) a 4-page book embedding of it.

the minimum value of k for which a k-page book embedding exists is commonly referred to as the *page number* (a.k.a. *book thickness* and *stack number*) of the graph. Accordingly, the page number of a graph family is defined as the maximum page number among its members.

In this context, a particularly intriguing research branch that has received significant attention over the years involves planar graphs and their various subclasses. While the page number of planar graphs has been recently resolved in the undirected setting [2, 23, 22], several open problems persist in the directed setting despite numerous efforts; see, e.g., [15]. Undoubtedly, the most notable one, which dates back to 1989 [18], is the one of specifying the page number of upward planar graphs; a sublinear upper bound was only recently proved in the literature [14]. In general, directed acyclic planar graphs may have unbounded page number as shown by Nowakowski and Parker [18]. This negative result was recently further strengthened by Jungeblut, Merker and Ueckerdt [15], who showed that even graphs with treewidth 2 may have unbounded page number. These negative results suggest that, in the directed setting, only very restricted subclasses of planar graphs are expected to have bounded page number. Indicatively, we mention the classes of directed trees [13], two-terminal series-parallel graphs [9], N-free graphs [16], directed acyclic outerpaths [17] and planar graphs whose faces have a special structure [5].

It is worth making a separate reference to the class of directed acyclic outerplanar graphs. For this elementary class, Heath, Pemmaraju and Trenk [13] back in 1999 conjectured a constant upper bound on their page number. Surprisingly enough, their conjectured remained open for more than twenty years. An important step towards settling it was done by Nöllenburg and Pupyrev [17] in 2023, who showed that the *monotone* directed acyclic outerplanar graphs have bounded page number; these graphs are constructed starting from a single so-called *base edge* by iteratively connecting a new vertex to the endpoints of an existing edge on the outer face using either two incoming or two outgoing edges incident to it (and thus, they are by definition maximal). More precisely, an explicit upper bound of 128 can be derived by combining the work by Nöllenburg and Pupyrev [17] with a known result by Davies [8]. Based on this result, Jungeblut, Merker and Ueckerdt [15] managed to settle in the positive the conjecture by Heath, Pemmaraju and Trenk [13]. In particular, they first observed the following relationship between the page number of monotone and general directed acyclic outerplanar graphs:

▶ **Theorem 1** (Jungeblut, Merker and Ueckerdt [15]). *If s is an upper bound on the page number of directed acyclic monotone outerplanar graphs, then $8 \cdot (12 \cdot (2s + 2) + 1)$ is an upper bound on the page number of general directed acyclic outerplanar graphs.*

Theorem 1 together with the aforementioned bound of 128 by Nöllenburg and Pupyrev [17], enabled Jungeblut, Merker and Ueckerdt [15] to prove that the page number of directed acyclic outeplanar graphs is at most 24,776. Of course, they asked for improvements to this bound.

Our Contribution. In this work, we present the first such improvement. In view of the multiplicative factors of the relationship given in Theorem 1, we naturally turn our attention to the class of directed acyclic monotone outerplanar graphs, for which we lower the upper bound on their page number from 128 [8, 17] to 5; as a side result of independent interest, we also present a corresponding lower bound of 4. A notable consequence of our upper bound is that, when it is coupled with Theorem 1, it yields a significant improvement of the upper bound on the page number of general directed acyclic outerplanar graphs from 24,776 to 1,160. A summary of our findings is given in the following theorem and its corollary.

▶ **Theorem 2.** *The page number of directed acyclic monotone outerplanar graphs is* 4 *or* 5.

▶ **Corollary 3.** *The page number of directed acyclic outerplanar graphs is at most* 1,160.

2 Preliminaries

In this paper, we consider *directed graphs*, i.e., graphs in which every edge e between two vertices u and v has an orientation, either from u to v, or from v to u; in the former case, we denote e by (u, v), while in the latter case by (v, u). When neglecting the orientation of e, we may refer to it with $\{u, v\}$. An edge in a directed graph is *transitive* if the graph contains a directed path of length at least 2 from its source to its target. A directed graph is *acyclic* if it contains no directed cycle. It is well-known that every directed acyclic graph admits a topological order.

A *linear order* $\prec$ of a directed acyclic graph G is a topological order of its vertices. For any two vertices u and v of G, we write $u \prec v$ if and only if (u, v) belongs to G and u precedes v in the topological order. We say that two independent edges (u, v) and (w, z) with $u \prec v$, $w \prec z$ and $u \prec w$ *cross* in $\prec$ if and only if $u \prec w \prec v \prec z$ holds. A *page* is a set of pairwise non-crossing edges with respect to $\prec$. A *k-page book embedding* of G consists of a linear order $\prec$ of its vertices and a partition of its edges into k pages with respect to $\prec$. The *page number* of G is the minimum value of k such that G admits a k-page book embedding.

Our focus is on *directed acyclic maximal outerplanar graphs*, which admit planar drawings with all vertices being incident to the outer face, and no edge can be added such that the resulting graph is still directed acyclic outerplanar. Such a graph is *monotone* if it can be defined through the following construction sequence: **(i)** A single edge (u, v) is a monotone directed acyclic outerplanar graph; we refer to this first edge in the construction sequence as *base edge*. **(ii)** If H is a monotone directed acyclic outerplanar graph and (x, y) is an edge of H incident to its outer face, then the graph obtained from H by adding a new vertex z and either the edges (x, z) and (y, z) or the edges (z, x) and (z, y) is again a monotone directed acyclic outerplanar graph; in both cases, we say that z is *stacked* on the edge (x, y). By the next lemma, we assume that the base edge of G is incident to the outer face.

▶ **Lemma 4.** *Every directed acyclic monotone outerplanar graph G with at least three vertices contains exactly two base edges that are incident to its outer face.*

Proof. If G has exactly three vertices, then its two non-transitive edges are base edges; see Figures 2a and 2b. So, assume that G has more than three vertices. Let z be the last vertex in the construction sequence of G and let (x, y) be the edge on which z is stacked. It follows that z is incident to exactly two edges, namely, either the edges (z, x) and (z, y), or the edges (x, z) and (y, z). We consider the case in which the edges (x, z) and (y, z) belong to G; the case in which the edges (z, x) and (z, y) belong to G is symmetric. Let H be the graph

obtained by removing z from G. Since z was the last vertex in the construction sequence of G, it follows that H is a directed acyclic monotone outerplanar graph. Hence, by induction, we may assume that H contains exactly two base edges that are incident to its outer face.

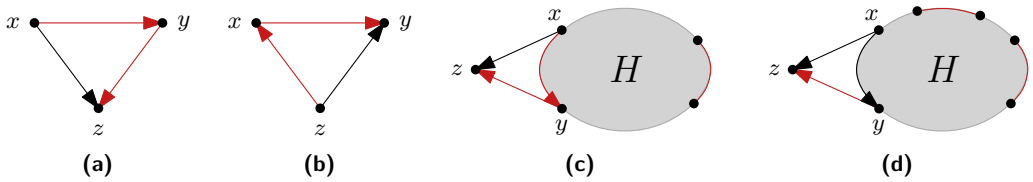


Figure 2 Illustrations for the proof of Lemma 4.

We distinguish two cases depending on whether (x, y) is a base edge of H or not. Note that, by definition of H, edge (x, y) is on the outer face of H. Assume first that (x, y) is indeed a base edge of H. Since (x, y) is a base edge of H, it follows that the edge (y, z) is a base edge of G; see Figure 2c. Since z is the last vertex in the construction sequence of G, since the edge (x, y) is an inner edge of G and since H has exactly two base edges on its outer face, it follows that G has exactly two base edges on its outer face. To complete the proof, assume that (x, y) is not a base edge of H. In this case, we prove that neither (x, z) nor (y, z) is a base of G, proving that G has exactly two base edges on its outer face, namely, the two of H. Since the edge (x, z) cannot be a base edge of G, assume for a contradiction that (y, z) is a base edge of G; see Figure 2d. It follows that there is a construction sequence that starting from (y, z) yields G. Neglecting z, this sequence implies a construction sequence for H, contradicting the fact that H has exactly two base edges on its outer face. ◀

▶ Remark 5. Note that there is a technical difference in the definition of directed acyclic monotone outerplanar graphs used in [15] and [17]. The latter requires the base edge to be on the outer face, while the former relaxes this requirement. In view of Lemma 4, the two definitions become equivalent. A notable implication of this equivalence is that substituting the twist number of 4 into the expression given by Davies [8] yields an upper bound of 64 for the page number of directed acyclic monotone outerplanar graphs (rather than 128, as used in [15]).

3 Embedding directed acyclic monotone outerplanar graphs in 5 pages

In the following, we describe how to embed a directed acyclic monotone outerplanar graph $G = (V, E)$ into five pages. Our approach consists of three main steps.

1. We define a *canonical* construction sequence for G that yields a rooted spanning tree T of the underlying undirected graph of G (see Section 3.1).
2. Based on these, we specify the linear order $\prec$ of the vertices of G, such that the edges of T do not cross in $\prec$ (see Section 3.2).
3. The remaining edges of G, namely, those that do not belong to T, can be 4-colored, such that the edges of the same color do not cross in $\prec$ (see Section 3.3).

It should be emphasized that T is rooted, but the orientation of the tree edges T in G does not reflect this hierarchy. Since T is rooted and spanning, we refer to a subtree of T rooted at a vertex v by $T(v)$ and to the set of non-tree edges by $N = E \setminus E(T)$.

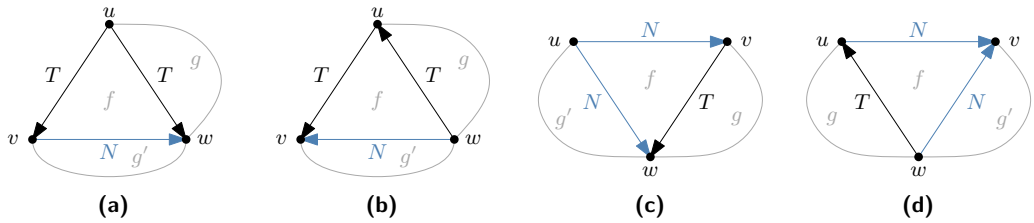


Figure 3 Illustrations for the definition of the canonical construction sequence.

3.1 A canonical construction sequence

Even though G has exactly two base edges incident to its outer face, there may exist several construction sequences that result in G, even if the starting base edge is fixed. We next describe a so-called *canonical* construction sequence that is uniquely defined if we fix the starting base edge to one of the two base edges incident to the outer face of G. To do so, we consider the weak dual G^* of G, which has a face-vertex u_f for each bounded face f of G and an edge between two vertices u_f and u_g if and only if the corresponding faces f and g of G are adjacent. G^* is a tree, which we assume to be rooted at the bounded face of G having the fixed base edge of G on its boundary. Since the fixed base edge is incident to the outer face of G, this face is uniquely defined. Further, since G is maximal, G^* is binary.

Starting from the root of G^*, we perform a specific DFS traversal of the vertices of G^*. When visiting a new vertex u_f of G^*, we assume that the edge (u, v) of G corresponding to the edge connecting u_f with its parent u_p in G^* has been assigned to T or to N. At the root of G^*, we assume that this edge is the fixed base edge which is assigned to T, such that its source is the parent of its target in T. The task is to assign the other two edges of f to T and N, and guide the traversal of G^* to each of the (at most) two subtrees of G^* rooted at u_f.

Let w be the third vertex of f. Assume first that the edge (u, v) belongs to T; see Figures 3a and 3b. Since (u, v) belongs to T, it follows that either u is the parent of v or v is the parent of u in T; recall that the edge orientations do not reflect the hierarchy in T. Consider first the case, in which f contains the edges (u, w) and (v, w). If u is the parent of v in T, then we assign the edge (u, w) to T and the edge (v, w) to N. We further assume that w is a child of u in T. On the other hand, if v is the parent of u in T, then we assign the edge (v, w) to T and the edge (u, w) to N. We further assume that w is a child of v in T. Consider now the case, in which f contains the edges (w, u) and (w, v). If u is the parent of v in T, then we assign the edge (w, u) to T and the edge (w, v) to N. We further assume that w is a child of u in T. On the other hand, if v is the parent of u in T, then we assign the edge (w, v) to T and the edge (w, u) to N. We further assume that w is a child of v in T. This completes the case where edge (u, v) belongs to T. Observe that in all four cases, we assigned to T the edges towards the vertex that is the parent among u and v.

Assume now that the edge (u, v) belongs to N; see Figures 3c and 3d. Consider first the case, in which f contains the edges (u, w) and (v, w). In this case, we assign (v, w) to T and (u, w) to N. We further assume that w is a child of v in T. Otherwise, f contains the edges (w, u) and (w, v) and we assign (w, u) to T and (w, v) to N. We further assume that w is a child of u in T. Observe that in both cases we assigned to N the transitive edge of face f.

So far, we have assigned the edges of f that are different from (u, v) to T and N. We next describe how the traversal of G^* proceeds. Let u_g and $u_{g'}$ be the (at most) two children of u_f in G^*, such that f and g share the edge assigned to T, while f and g' share the edge assigned to N in G. We recursively traverse the subtree of G^* rooted at u_g starting from u_g and then recursively traverse the subtree of G^* rooted at $u_{g'}$ starting from $u_{g'}$.

The procedure described above uniquely defines a canonical construction sequence $\pi = \{v_1, v_2, \ldots, v_n\}$ of G, where (v_1, v_2) is the fixed starting base edge of G. For $i \geq 3$, let G_i be the subgraph of G induced by $v_1, \ldots, v_i$. It follows that vertex v_{i+1} is connected with exactly two (neighboring) vertices in G_i, while the edges realizing these connections have been assigned to T and N. If the edge between the neighbors of v_{i+1} in G_i belongs to N, then the non-tree edge incident to v_{i+1} is transitive. The fact that T is a spanning tree of G follows by construction: each time that we define the next vertex v_{i+1} in the canonical construction sequence exactly one of the two edges connecting v_{i+1} with its two neighbors (incoming or outgoing) in G_i is assigned to T; that is, v_{i+1} is added as a leaf to T. We summarize these observations below and then present two properties of the canonical construction sequence.

T.1 The edges of T induce a spanning tree of G

T.2 The root of T is the source of the base edge of G.

T.3 Vertex v_{i+1} is a leaf in the restriction of T to G_{i+1}.

▶ **Property 1.** *Assume that the edge between the neighbors v_c and v_p of v_{i+1} in G_i belongs to T, such that v_p is the parent of v_c in T. Then $i = c$ holds, that is, v_{i+1} and v_c are consecutive in the canonical construction sequence.*

Proof. We argue along the traversal of the dual G^*. Let f be the face of G bounded by v_{i+1}, v_p and v_c. Consider the parent vertex u_g of u_f in G^*. Then, faces f and g share the edge between v_p and v_c. Since this edge belongs to T, it follows that once the visit of u_g is completed, the traversal of G^* will immediately continue to u_f. In other words, after face g the third vertex of face f is appended to the canonical construction sequence. This implies that v_c and v_{i+1} are consecutive in the canonical construction sequence. Hence, $i = c$. ◀

▶ **Property 2.** *Let (v_k, v_l) be a non-tree edge of G, that is, $(v_k, v_l) \in N$. Then, the endpoints of (v_k, v_l) belong to disjoint subtrees, that is, $T(v_k) \cap T(v_l) = \emptyset$ holds.*

Proof. We argue by induction on the length of the canonical construction sequence. The property holds in G_2 (i.e., for the base case $i = 2$), since $N = \emptyset$. Assume that for all non-tree edges of G_i with $2 < i < n$ the property holds. We prove that the property holds in G_{i+1}. Consider vertex v_{i+1}. We distinguish two cases based on whether the edge between the neighbors of v_{i+1} in G_i belongs to T or to N. In the former case, let v_p and v_c be the neighbors of v_{i+1} in G_{i+1}, such that v_p is the parent of v_c in T. Then, the edge between v_{i+1} and v_c is the non-tree edge incident to v_{i+1}. This edge connects to two siblings of v_p in T. Therefore, the endpoints of this edge belong to disjoint subtrees, namely, $T(v_{i+1}) \cap T(v_c) = \emptyset$. Consider now the case where the edge between the neighbors of v_{i+1} in G_i belongs to N. Let v_p and v_c be the neighbors of v_{i+1} in G_{i+1} and assume that v_{i+1} becomes a child of v_p in G_{i+1}. The edge between v_{i+1} and v_c is the non-tree edge incident to v_{i+1} in G_{i+1}. By induction hypothesis the endpoints of the edge between v_c and v_p are in disjoint subtrees. Since $v_{i+1} \in T(v_p)$, the same holds for the endpoints of the edge between v_{i+1} and v_c, i.e., $T(v_{i+1}) \cap T(v_c) = \emptyset$. ◀

3.2 Computing the linear order

We next describe the linear order $\prec$ of G. A key ingredient in our approach is the so-called *consecutive subtree property*, which requires for every vertex v in G the vertices in the subtree $T(v)$ to appear consecutively in $\prec$. We denote the corresponding interval of $\prec$ induced by $T(v)$ by $I_{\prec}(v)$. Note that by definition v is part of $I_{\prec}(v)$. To guarantee this property, for $2 \leq i < n$, we assume that we have recursively computed a linear order $\prec_i$ of G_i, such that:

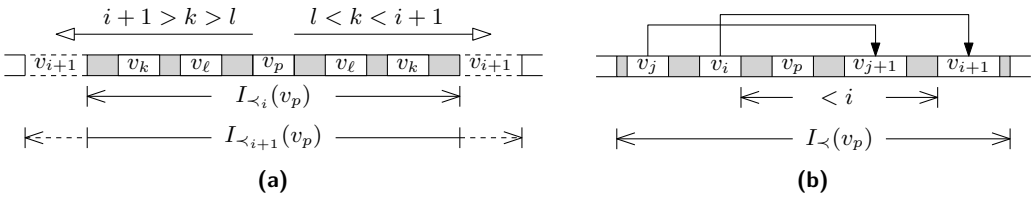


Figure 4 (a) Inserting v_{i+1} in $\prec_i$ for G_i to derive $\prec_{i+1}$ for G_{i+1}. (b) The case in which two sibling edge (v_i, v_{i+1}) and (v_j, v_{j+1}) with common parent v_p cross.

L.1 $\prec_i$ is a topological order of G_i

L.2 $\prec_i$ obeys the *consecutive subtree property* in G_i

L.3 Let v_p be the parent of two distinct children v_k and v_ℓ in T. Then, the following holds:

$$v_k \prec_i v_\ell \prec_i v_p \quad \text{or} \quad v_p \prec_i v_\ell \prec_i v_k \quad \Rightarrow \quad k > \ell.$$

In the base case $i = 2$, G_2 consists of the base edge (v_1, v_2), which is part of T; thus L.1–L.3 are trivially satisfied. For $i > 2$, we show how to insert v_{i+1} into $\prec_i$ to derive a linear order $\prec_{i+1}$ of G_{i+1} that satisfies L.1 - L.3; see Figure 4a. Assume that v_{i+1} is a child of v_p in the restriction of T in G_{i+1}. We consider two cases based on the orientation of the edge between v_{i+1} and v_p; recall that the edge orientations do not reflect the hierarchy in T. Assume first that this edge is the edge (v_p, v_{i+1}). We insert v_{i+1} directly after the last vertex of $I_{\prec_i}(v_p)$. Otherwise, we insert v_{i+1} directly before the first vertex of $I_{\prec_i}(v_p)$. This ensures that L.2 holds for G_{i+1}, since $I_{\prec_{i+1}}(v_p) = I_{\prec_i}(v_p) \cup \{v_{i+1}\}$ is consecutive. For L.3, we observe that for every child v_c of v_p in $I_{\prec_i}(v_p)$, it holds that $i + 1 > c$, thereby satisfying the invariant. It remains to show that $\prec_{i+1}$ is indeed a topological order of G_{i+1}. Let v_j be the neighbor of v_{i+1} in G_i that is different from v_p. We consider two cases based on whether the edge between v_j and v_p belongs to T or to N. In the former case, assume w.l.o.g. that the edges from v_{i+1} to v_j and v_p are incoming to v_{i+1}; the case, where these edges are outgoing, is symmetric. Since we inserted v_{i+1} after $I_{\prec_i}(v_p)$ and $v_j \in I_{\prec_i}(v_p)$, we have $v_j \prec_{i+1} v_{i+1}$ and $v_p \prec_{i+1} v_{i+1}$ as desired. Assume that the edge between v_j and v_p belongs to N. Since the edge between v_{i+1} and v_p belongs to T, the edge between v_j and v_{i+1} belongs to N and thus is transitive in the face formed by v_{i+1}, v_j and v_p. Again, assume w.l.o.g. that the edges from v_{i+1} to v_j and v_p are incoming to v_{i+1}; the case, where these edges are outgoing, is symmetric. In this case, it follows that $v_p \prec_{i+1} v_{i+1}$ holds. By transitivity of (v_j, v_{i+1}), we get that (v_j, v_p) exists. Hence, $v_j \prec_{i+1} v_p \prec_{i+1} v_{i+1}$ holds and L.1–L.3 are satisfied by $\prec_{i+1}$.

We next present three key properties of the linear order $\prec$ of G. For any two vertices u and v of G, we write $\mathrm{LCA}(u, v)$ to denote the lowest common ancestor of u and v in T.

▶ **Property 3.** *For every two vertices u and v of G, $u \in I_\prec(v)$ if and only if $u \in T(v)$.*

Proof. It follows from the fact that $\prec$ obeys the consecutive subtree property (L.2). ◀

▶ **Property 4.** *Let (u, v) and (w, z) be two edges that cross in $\prec$. Then, both of the following two conditions hold:* **(i)** *u or v is in $T(LCA(w, z))$ and* **(ii)** *w or z is in $T(LCA(u, v))$.*

Proof. W.l.o.g. assume that $u \prec w \prec v \prec z$ holds. Since (w, z) belongs to G and since $w \prec v \prec z$, by L.2 we obtain $v \in I_\prec(\mathrm{LCA}(w, z))$. Symmetrically, $w \in I_\prec(\mathrm{LCA}(u, v))$. Then, the property follows from Property 3. ◀

Notice that Property 4 holds for all edges, including edges of T. However, for a pair of two independent tree edges only one of the two conditions can be satisfied, which directly implies the following property guaranteeing that all edges of T fit into a single page as desired.

▶ **Property 5.** *No two edges of T cross in $\prec$.*

3.3 Assigning edges to pages

The overall approach for the edge assignment is as follows. We dedicate a single page for edges belonging to the tree T (by Property 5). The algorithm for coloring the non-tree edges is notably simple (though the proof of correctness will be tedious). More precisely, the non-tree edge incident to the next vertex v_{i+1} ($i = 3, \ldots, n-1$) in the canonical construction sequence is colored either **(i)** with the color that is not used by its two neighbors in G_i, if v_{i+1} is stacked on a tree edge, or **(ii)** with the color of the non-tree edge it is stacked on, otherwise. In the following subsections, we will prove that this coloring yields a valid 5-page book embedding.

3.3.1 Partitioning the non-tree edges to groups

For the non-tree edges in N, we will first partition them into groups. These groups will then be 4-colored such that groups of the same color can be assigned to the same page, i.e., no two edges of the same color cross in $\prec$. In order to form the desired partition of N, we need a few more notions. We start with the notion of *sibling edge*. An edge (v_i, v_j) that belongs to N with v_i and v_j having the same parent v_p in T is called a *sibling edge* with *parent* v_p. Sibling edges satisfy the following property in the canonical construction sequence.

▶ **Property 6.** *If (v_i, v_j) is a sibling edge with parent v_p, then $i = j + 1$ or $j = i + 1$ holds.*

Proof. Follows directly from Property 1 of the canonical construction sequence. ◀

Using Property 6, we next prove that no two sibling edges with the same parent cross in $\prec$.

▶ **Property 7.** *If $\{v_i, v_{i+1}\}$ and $\{v_j, v_{j+1}\}$ are two sibling edges with common parent v_p, then they do not cross in $\prec$.*

Proof. Assume to the contrary that they cross; see Figure 4b. We distinguish cases based on the relative order of the endpoints of these edges with respect to the common parent v_p. Let us first consider the case where the endpoints of at least one of the two edges either both precede v_p or follow v_p. W.l.o.g. assume v_p follows both v_i and v_{i+1} in $\prec$. By L.3, we have $v_{i+1} \prec v_i \prec v_p$. Since both v_j and v_{j+1} are children of v_p and $j \neq i$, neither of them can be between v_i and v_{i+1} in $\prec$. Therefore, $\{v_i, v_{i+1}\}$ and $\{v_j, v_{j+1}\}$ cannot cross. We conclude that in order for $\{v_i, v_{i+1}\}$ and $\{v_j, v_{j+1}\}$ to cross, the endpoints of each of these two edges have to be on different sides of v_p in $\prec$, that is, v_p is between the endpoints of each of these edges in $\prec$. Since $\{v_i, v_{i+1}\}$ and $\{v_j, v_{j+1}\}$ cross, they are independent and therefore $i + 1 \neq j$. Assume w.l.o.g. that $i + 1 < j$. In the following, we focus on the case where $v_i \prec v_p \prec v_{i+1}$; the case in which $v_{i+1} \prec v_p \prec v_i$ is symmetric. From L.3 we get that for every child v_k of v_p with $v_i \prec v_k \prec v_{i+1}$, we have $k < i$. For $\{v_i, v_{i+1}\}$ and $\{v_j, v_{j+1}\}$ to cross either v_j or v_{j+1} has to be between v_i and v_{i+1} in $\prec$ contradicting that $i + 1 < j$. ◀

So far we have shown that tree edges can fit together in a single page and that the same holds for sibling edges having the same parent. Two questions that arise at this point are how to handle sibling edges with different parents and non-tree edges that are not sibling

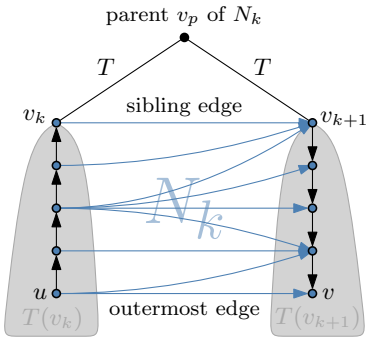


Figure 5 Illustration for Invariants N.1–N.3.

edges. For the latter, the idea is to assign these edges to *groups* such that each group has a designated sibling edge as *representative*. As we will see, a crossing between two non-tree edges of different groups can be reduced to a crossing between the groups' representatives.

Let us formally introduced the concept of groups. As already mentioned, each *group* of non-tree edges has exactly one representative, which is a sibling edge. In other words, groups and sibling edges are in a one-to-one correspondence. Further, a sibling edge belongs to the group it represents. By Property 6, we denote with $N_i \subset N$ the group that is represented by the sibling edge $\{v_i, v_{i+1}\}$. It follows that $\{v_i, v_{i+1}\} \in N_i$. Note that the index i stems from the rank of v_i and v_{i+1} in the canonical construction sequence π. Since not every pair of consecutive vertices in π yields a sibling edge, it follows that there does not exist a group N_i for every integer i in $[1, n-1]$. We further require that the groups form a partition of N.

We now describe how to obtain such a partition of N. We do so by using the canonical construction sequence and by imposing further invariant properties that will help us later to show that no more than four colors are needed in order to color the groups that we will form. With this in mind, we also need to ensure that non-tree edges of the same group can always be assigned to the same page. We capture these properties with the following invariants.

Assume that all non-tree edges of G_i with $2 < i < n$ have been partitioned to groups $N_1, \ldots, N_{i-1}$, such that the following hold for each non-empty group N_k with $k \in [1, i-1]$:

N.1 N_k has exactly one representative sibling edge $\{v_k, v_{k+1}\}$ of G_i.

N.2 For every edge (u, v) of G_i that belongs to group N_k one of the following holds:

- **a.** If (v_k, v_{k+1}) is the representative of N_k, then there is a directed path in $T(v_k)$ from u to v_k and a directed path in $T(v_{k+1})$ from v_{k+1} to v.
- **b.** If (v_{k+1}, v_k) is the representative of N_k, then there is a directed path in $T(v_{k+1})$ from u to v_{k+1} and a directed path in $T(v_k)$ from v_k to v.

Each vertex of these directed paths is incident to at least one edge of N_k.

N.3 Each non-empty group N_k is also associated with an *outermost edge* (u, v), which is the solely edge on the outer face of G_i among the edges of N_k and *covers* all edges of N_k in $\prec$, that is, for each $(u', v') \in N_k$, we have $u \preceq u' \preceq v' \preceq v$.

We prove that the aforementioned invariants hold also for the non-tree edges of G_{i+1} using the following approach. Consider vertex v_{i+1}. We distinguish cases based on whether the edge between the neighbors of v_{i+1} in G_i belongs to T or to N.

We first consider the case in which the edge between the neighbors of v_{i+1} in G_i belongs to T. Let v_p and v_c be these neighbors, such that v_p is the parent of v_c in T. In this case, vertex v_{i+1} is a child of v_p in G_{i+1}, that is, the edge between v_{i+1} and v_p belongs to T, while the edge between v_{i+1} and v_c belongs to N. In particular, the edge between v_{i+1} and v_c is a

sibling edge. We assign it to a new group N_i and we make it both the representative and the outermost of this group. In other words, in this case we are creating a new group that solely consists of a single sibling edge. N.1–N.3 are satisfied because the edge between v_{i+1} and v_c cannot be associated with any of the groups N_k with $k \in [1, i-1]$, belongs to the outer face of G_{i+1} and trivially covers all edges of N_i, as there are no other edges in this group.

Consider now the case where the edge between the neighbors of v_{i+1} in G_i belongs to N. Let v_l and v_r be these neighbors. To simplify the presentation, we assume that the edge between v_l and v_r is oriented from v_l to v_r; the other case is symmetric. Since (v_l, v_r) belongs to G_i, by N.1 the edge (v_l, v_r) has been assigned to a group, say N_j with $j \in [1, i-1]$, whose representative edge is (v_j, v_{j+1}) by N.2. Assume that the edges connecting v_{i+1} with v_l and v_r in G_{i+1} are outgoing from v_{i+1}; the case in which the edges connecting v_{i+1} with v_l and v_r in G_{i+1} are incoming to v_{i+1} is symmetric. Under this assumption, the edge (v_{i+1}, v_r) is the non-tree edge incident to v_{i+1} in G_{i+1}. We proceed by assigning this edge to N_j. Regarding N.1 there is nothing to be proven, as N_j is an existing group. Since (v_j, v_{j+1}) is the representative of N_j, we know that by N.2 applied on G_i that there exist a directed path in $T(v_j)$ from v_l to v_j and a directed path in $T(v_{j+1})$ from v_{j+1} to v_r. Since v_{i+1} has an outgoing edge to v_l, it follows that there is a directed path from v_{i+1} to v_l, as well, guaranteed that N.2 is satisfied from G_{i+1}. To prove that N.3 is satisfied in G_{i+1}, we first observe that (v_l, v_r) is the outermost edge of N_j, since it is the one incident to the outer face of G_i among the edges of N_j by N.2. As a result, it covers all edges of G_i belonging N_j in $\prec$. In G_{i+1}, the edge (v_{i+1}, v_r) becomes that outermost edge of N_j, as it is the solely edge of N_j incident to the outer face of G_{i+1}. Furthermore, (v_{i+1}, v_r) covers all edges of G_{i+1} belonging N_j in $\prec$, since v_{i+1} appear right before v_l in $\prec$. This guarantees that N.3 is satisfied in G_{i+1}.

3.3.2 Properties of the formed groups

Having proved that N.1–N.3 hold for the non-tree edges of G_{i+1}, and thus by induction of G, we proceed to investigate properties of the groups that are formed with this approach.

▶ **Property 8.** *No two edges assigned to the same group cross in $\prec$.*

Proof. Assume that no two (non-tree) edges of the same group cross in $\prec$ for the subgraph G_i of G. By Invariant N.3, it follows that the non-tree edge incident to v_{i+1} in G_{i+1} covers all edges of the same group in G_i, as it becomes the outermost of this group. This implies that no two edges of the same group cross in $\prec$ for the subgraph G_{i+1} of G. This completes the proof, since in G_2 the property trivially holds. ◀

▶ **Property 9.** *Let N_i and N_j be two distinct groups with group parents v_k and v_l, respectively, such that neither v_k is an endpoint of an edge in N_j nor v_l is an endpoint of an edge in N_i. If $v_k \neq v_l$ or if the representative sibling edges of N_i and N_j are independent, then no two edges in N_i and N_j cross in $\prec$.*

Proof. Let (u, v) and (x, y) be two edges of N_i and N_j, respectively. Let also $\{v_i, v_{i+1}\}$ and $\{v_j, v_{j+1}\}$ be the representative sibling edges of N_i and N_j, respectively. We assume that the former edge is oriented from v_i to v_{i+1}, while the latter from v_j to v_{j+1}; the remaining cases are handled symmetrically. It follows by N.2 that $u \in T(v_i)$, $v \in T(v_{i+1})$, $x \in T(v_j)$ and $y \in T(v_{j+1})$. Since v_k and v_l are the group parents of N_i and N_j, these relationships imply that $u, v \in T(v_k)$ and $x, y \in T(v_l)$. Assume to the contrary that (u, v) and (x, y) cross in $\prec$. Consider first the case where the two subtrees $T(v_k)$ and $T(v_l)$ are vertex-disjoint. L.2 implies that both u and v either precede or follow both x and y in $\prec$. This implies that (u, v) and (x, y) do not cross in $\prec$; a contradiction. Hence, $T(v_k)$ and $T(v_l)$ are not vertex-disjoint.

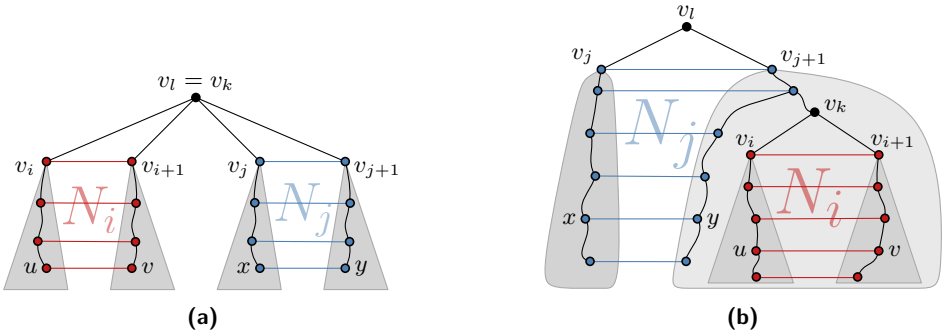


Figure 6 Illustrations for the proof of Property 9: (a) $T(v_k)$ and $T(v_l)$ share the same root; (b) $T(v_k)$ is a proper subtree of $T_(v_l)$.

Assume first that $T(v_k)$ and $T(v_l)$ share the same root, that is, $v_k = v_l$; see Figure 6a. It follows by the assumptions of the lemma that the representative sibling edges (v_i, v_{i+1}) and (v_j, v_{j+1}) of N_i and N_j are independent, implying that the subtrees $T(v_i)$, $T(v_{i+1})$, $T(v_j)$ and $T(v_{j+1})$ are vertex-disjoint. By N.2, we know that the vertices of each of these four subtrees are consecutive in $\prec$. Since (u, v) and (x, y) cross in $\prec$, it follows that either $u \prec x \prec v \prec y$ or $x \prec u \prec y \prec v$. W.l.o.g. consider the former case; the latter follows by the same argument. Since $u \in T(v_i)$, $v \in T(v_{i+1})$, $x \in T(v_j)$ and $y \in T(v_{j+1})$ and since the vertices of each of these four subtrees are consecutive in $\prec$, it follows that $v_i \prec v_j \prec v_{i+1} \prec v_{j+1}$, that is, the representative edges of N_i and N_j cross in $\prec$; a contradiction to Property 7.

Hence, we deduce that $v_k \neq v_l$. Since $T(v_k)$ and $T(v_l)$ are not vertex-disjoint and since $v_k \neq v_l$, it follows that either $T(v_k)$ is a proper subtree of $T(v_l)$ or vice versa. Assume the former, that is, $T(v_k) \subsetneq T(v_l)$; see Figure 6b. Since the vertices of $T(v_k)$ appear consecutively in $\prec$, it follows that in order for (u, v) and (x, y) to cross in $\prec$, one endpoint of (x, y), say w.l.o.g. x, has to be contained in $T(v_k)$. Then, by N.2, there exists a path between x and v_l in T, such that with the exception of v_l all vertices on this path are incident to an edge of N_j. Since x belongs also to $T(v_k)$, this path necessarily contains vertex v_k, which is a contradiction to the fact that v_k is not an endpoint of an edge in N_j. ◀

3.3.3 The groups can be 4-colored

We next color the groups with four colors, such that edges in groups with the same color do not cross in $\prec$ (recall Property 9), thus completing the proof of the upper bound of Theorem 2.

▶ **Property 10.** *There exists a 4-coloring of the groups that contain the non-tree edges of G, such that no two edges assigned to two groups of the same color cross in $\prec$.*

Proof. Our proof is by induction on the length of canonical construction sequence. Assume that we have computed a 4-coloring of the non-tree edges of G_i satisfying the next invariants:

- **C.1** Edges of the same group in G_i are of the same color.
- **C.2** Each vertex in G_i is incident to at most two edges belonging to groups of different colors, that is, the non-tree edges incident to it are either single-colored or bicolored.
- **C.3** For every tree edge $\{v_p, v_c\}$ on the outer face of G_i, with v_c being a child of v_p, all non-tree edges incident to v_c belong to the same group, that is, they are single-colored.
- **C.4** If N_k and N_l are two distinct groups with group parents v_p and v_q, such that either v_p is an endpoint of an edge in N_l or v_q is an endpoint of an edge in N_k, then N_k and N_l are of different colors.



Figure 7 Illustration for two main cases of the proof of Property 10, in which the edge between neighbors of v_{i+1} in G_i is: (a) a non-tree edge (v_l, v_r) belonging to N_j, (b) the tree edge $\{v_p, v_i\}$, whereas $\{v_i, v_{i+1}\}$ is the representative sibling edge of the new group N_i.

C.5 If N_k and N_l are two distinct groups with common group parent v_p, such that their representative sibling edges $\{v_k, v_{k+1}\}$ and $\{v_l, v_{l+1}\}$ share an endpoint, then N_k and N_l are of different colors.

Clearly, for G_2 all invariants are satisfied since $N = \emptyset$. So, we assume that $i > 2$ and we will prove that we can appropriately color the edges incident to v_{i+1} in G_{i+1}, such that C.1–C.5 are satisfied by for the non-tree edges in G_{i+1}. We distinguish cases based on whether the edge between the neighbors of v_{i+1} in G_i belongs to T or to N, that is, v_{i+1} is stacked on a tree or a non-tree edge of G_i, respectively.

Stacking on a non-tree edge. Assume first that the edge, say (v_l, v_r), between the neighbors v_l and v_r of v_{i+1} in G_i belongs to some group N_j; see Figure 7a. By C.1, we can assume that the color assigned to the edges of N_j is w.l.o.g. c_1. To simplify the presentation, we assume that the two edges incident to v_{i+1} in G_{i+1} are outgoing from v_{i+1}, that is, (v_{i+1}, v_l) and (v_{i+1}, v_l) belong to G; the case of these edges being incoming is symmetric by exchanging the roles of v_l and v_r. Under this assumption, the non-tree edge incident to v_{i+1} in G_{i+1} is the edge (v_{i+1}, v_r); the edge (v_{i+1}, v_r) is the tree edge incident to v_{i+1} in G_{i+1}. Since (v_{i+1}, v_r) belongs to N_j, we color it with color c_1, so as to guarantee C.1. Since the set of colors that is used for the edges incident to v_r and v_l does not change and since there is only one non-tree edge incident to v_{i+1} in G_{i+1} (namely, the edge (v_{i+1}, v_r)), it follows that C.2 is guaranteed in G_{i+1}. Regarding C.3, we observe that the tree edge (v_{i+1}, v_r) incident to v_{i+1} in G_{i+1} is an edge on the outer face of G_{i+1}. Since for the tree edge (v_{i+1}, v_r) v_{i+1} is the child of v_l in T and since there is only one non-tree edge incident to v_{i+1} in G_{i+1} (namely, the edge (v_{i+1}, v_r)), it follows that C.3 is maintained in G_{i+1}. Since v_{i+1} is a leaf in the restriction of T to G_{i+1}, C.4 is satisfied by the inductive hypothesis. By the same argument and the fact that no new sibling edges are introduced in G_{i+1}, C.5 holds in G_{i+1}. This completes the case where the edge between the neighbors of v_{i+1} in G_i belongs N.

Stacking on a tree edge. We proceed to the more tedious case, where the edge between the neighbors of v_{i+1} in G_i belongs to T; see Figure 7b. Let v_p and v_c be the endpoints of this edge, such that v_p is the parent of v_c in T. By Property 1, $c = i$ holds. Also, by T.3, it follows that $v_c = v_i$ is a leaf in the restriction of T to G_i. In this case, the edge $\{v_i, v_{i+1}\}$ is the representative of the new group N_i in G_{i+1} solely consisting of $\{v_i, v_{i+1}\}$, while $\{v_p, v_{i+1}\}$ belongs to T. The former implies that regardless of the color that we will assign to N_i, C.1 holds in G_{i+1}. We further observe that since v_i and v_{i+1} are siblings and v_i is a leaf in the restriction of T to G_i, it remains a leaf also in the restriction of T to G_{i+1}. In particular, this implies that v_i is not the group parent of some group in G_{i+1}.

To guarantee C.4 and C.5, we have to choose the color for the representative sibling edge $\{v_i, v_{i+1}\}$ in N_i appropriately. In particular, we have to ensure that the color chosen is different from the colors incident to v_p and v_i in G_i. By C.2, the non-tree edges incident

to v_p in G_i have at most two colors, say c_1 and c_2. Since $\{v_p, v_i\}$ is incident to the outer face of G_i and since v_i is a child of v_p in T, by Item C.3 the non-tree edges incident to v_i in G_i belong to the same group; let c_3 be the color assigned to this group. The color that we assign to N_i is one not belonging in $\{c_1, c_2, c_3\}$, say c_4.

Invariant C.2 is trivially satisfied at v_{i+1} because of its degree in G_{i+1}. Since by C.3 the non-tree edges incident to v_i in G_i belong to a single group, C.2 is satisfied for v_i in G_{i+1}. The same holds for v_p by the inductive hypothesis, since the edge $\{v_p, v_{i+1}\}$ belongs to T. Hence, C.2 is maintained G_{i+1}. Since the edge $\{v_p, v_{i+1}\}$ belongs to the outer face of G_{i+1} and since v_{i+1} has only one non-tree edge incident to it in G_{i+1} (i.e., the edge $\{v_i, v_{i+1}\}$), C.3 is maintained in G_{i+1}. Note that the edge (v_p, v_i) is not on the outer face of G_{i+1}.

For C.4, consider a group N_k with $k < i$ and group parent v_q and assume that either v_p is an endpoint of an edge in N_k or v_q is an endpoint of an edge in N_i. For the latter case, note that since N_i consists only of its representative sibling edge $\{v_i, v_{i+1}\}$, the only possible candidate for v_q is v_i. However, v_i is a leaf in G_{i+1} and therefore not a parent of a group. For the former case, that is, there is a non-tree edge incident to v_p (the parent of N_i) having the same color as N_i, we argue with our definition of c_4 that this is not the case. For C.5 it is sufficient to consider a possible sibling edge sharing an endpoint with $\{v_i, v_{i+1}\}$. The only possible candidate in G_{i+1} would be a sibling edge $\{v_{i-1}, v_i\}$. However, we ensure that c_4 is different from the color of any other non-tree edge incident to v_i which is sufficient to maintain C.5 in G_{i+1}. This completes the proof that C.1–C.5 are maintained in G_{i+1}.

Putting everything together. To prove that no two non-tree edges assigned to two groups of the same color cross in $\prec$, we combine all properties that we have obtained so far. Let e and e' be two such edges and assume that e and e' belong to N_i and N_j and that v_p and v_q are the group parents of N_i and N_j, respectively. By C.1 and Property 8, it follows that if $N_i = N_j$, then e and e' do not cross in $\prec$, as desired. Hence, we may assume that N_i and N_j are two distinct groups. If v_p is an endpoint of an edge in N_i or v_q is an endpoint of an edge in N_j, then N_i and N_j are of different colors by C.4; a contradiction to the fact that e and e' belong to groups of the same color. It follows that neither v_p is an endpoint of an edge in N_i nor v_q is an endpoint of an edge in N_j. In this case, if $v_p \neq v_q$, then by Property 9, e and e' do not cross in $\prec$. Therefore, we may assume that N_i and N_j share the same parent, that is, $v_p = v_q$. If the representative sibling edges of N_i and N_j share an endpoint, then by C.5, it follows that N_i and N_j are of different colors; a contradiction. Hence, the representative sibling edges of N_i and N_j are independent. Then, again Property 9 applies which states that e and e' do not cross in $\prec$. This concludes the proof of the property, which in turn coupled with Property 5 concludes the proof of the upper bound of Theorem 2. ◀

3.4 Time complexity

Given an directed acyclic graph, we prove in the following lemma that we can determine whether it is monotone outerplanar and compute a 5-page book embedding of it, if so, in linear time.

▶ **Lemma 6.** *Given a directed acyclic graph, there is a linear time algorithm to determine whether it is monotone outerplanar and to compute a* 5*-page book embedding of it, in the positive case.*

Proof. Let G be a directed acyclic n-vertex graph. We first check in $O(n)$ time whether the underlying undirected graph of G is maximal outerplanar. If so, we assume a maximal outerplanar embedding and proceed to check whether G is monotone. In view of Lemma 4,

this can be trivially done in $O(n^2)$ time, since given an edge e on the boundary of G one can check whether G is monotone with e being a base edge in $O(n)$ time by performing a BFS traversal of the weak dual G^* of G starting from the bounded face having e on its boundary.

For an $O(n)$ implementation, let f be a face that corresponds to a leaf u_f of the weak dual G^* of G. If f is the solely bounded face in G, then, since G is directed acyclic, it follows that G is monotone and we can also report the two base edges on its boundary (as observed in Lemma 4). Otherwise, let f' be the bounded face neighboring f in G, which is uniquely defined, since u_f is a leaf in G^*. Denote by (x, y) the edge shared by f and f' in G, by z the third vertex of f (which is of degree 2, since u_f is a leaf in G^*), and by H the subgraph of G obtained by the removal of z from G. There exist two cases for G to be monotone; **(i)** either one of the two edges incident to z is a base edge of G, or **(ii)** none of these two edges is a base edge of G. In Case (i), the edges incident to z are the edges (x, z) and (z, y); further, the edge (x, y) must be a base edge of H. In Case (ii), the edges incident to z are either the edges (x, z) and (y, z), or the edges (z, x) and (z, y); further, the edge (x, y) cannot be a base edge of H. Now, we recursively check whether H is monotone. In the negative case, we report that G is not monotone, as well. In the positive case, we may assume that the recursion also reports the two base edges of H. With this information, we can check in constant time, whether G is monotone by just observing the orientation of the edges incident to z depending on whether the edge (x, y) is a base edge of H or not. Since each such check takes constant time and since we perform at most $n - 4$ such checks (that is, the number of bounded faces of G), it follows that in $O(n)$ time we can check whether G is monotone.

If G meets the requirement of being monotone outerplanar and with a base edge on the outer face given, we can easily construct the canonical construction sequence and the spanning tree T of G in $O(n)$ time by a DFS traversal of the weak dual G^* of G as described in Section 3.1. For the linear order $\prec$, one may use the canonical construction sequence and maintain for every vertex v_p two ordered lists of its children in T. The first list contains the children that have to precede v_p in $\prec$, while the second contains the ones that follow v_p in $\prec$ as described in Section 3.2. Once these lists have been obtained, a simple in-order traversal of T yields $\prec$. Therefore, $\prec$ can also be computed in $O(n)$ time. As a final step, the coloring of the non-tree edges (that is, their page assignment) can be trivially done in $O(n)$ time by the simple approach described in the introduction of Section 3.3. This concludes the description of the linear-time implementation. ◀

4 Lower bounds

In this section, we first demonstrate a relatively small directed acyclic monotone outerplanar graph and prove combinatorially that it does not admit a book embedding with two pages.

▶ **Theorem 7.** *There exist directed acyclic monotone outerplanar graphs that do not admit book embeddings in two pages.*

Proof. The graph given in Figure 8a consists of nine vertices $x, y, v_1, \ldots, v_7$ and it is indeed a directed acyclic monotone outerplanar graph, whose base edge is the edge (v_1, v_3). Assume for a contradiction that there exists a book embedding $\mathcal{L}$ of it with two pages p_0 and p_1. Since the graph contains the directed path $v_1 \to \ldots \to v_7$, these vertices appear in this order in $\mathcal{L}$. Furthermore, w.l.o.g., we may assume that the edges (v_a, v_{a+2}) are assigned to page p_0, if a is even, and to page p_1, if a is odd (see the blue and red edges in Figure 8b).

Due to the edges (v_4, x) and (y, v_4), vertices x and y have to appear after and before vertex v_4 in $\mathcal{L}$, respectively. In particular, vertex x cannot appear between v_4 and v_6 in $\mathcal{L}$, as otherwise the edge (v_2, x) crosses (v_1, v_3) and (v_4, v_6), which are assigned to p_0 and p_1,

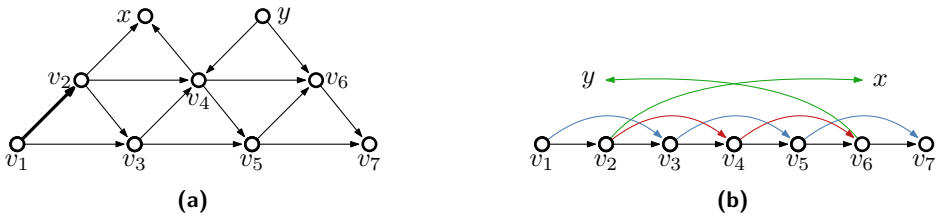


Figure 8 A directed acyclic monotone outerplanar graph not admitting a 2-page book embedding.

respectively. Symmetrically, vertex y cannot appear between v_2 and v_4. This implies that (v_2, x) and (y, v_6) cross in $\mathcal{L}$, which, in turn, implies that they have to be assigned to different pages. If (v_2, x) is assigned to p_0, then it crosses (v_1, v_3), which is also in p_0; a contradiction. If (y, v_6) is assigned to p_0, then it crosses (v_5, v_7), which is also in p_0; a contradiction. It follows that there is no 2-page book embedding for the graph, completing the proof. ◀

▶ Remark 8. We remark that there exist directed acyclic monotone outerplanar graphs that do not admit book embeddings with three pages; for an example refer to Figure 9. This statement is the outcome of the application of a well-known approach [3], which using SAT solving allows to test whether a given graph can be embedded in a book with a certain number of pages. Note that we are not aware of a directed acyclic monotone outerplanar graph that cannot be embedded in 4 pages. This motivated the title of our paper.

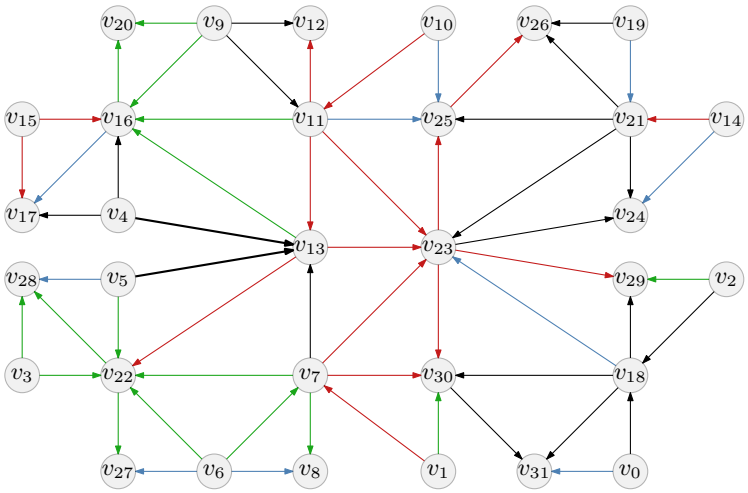


Figure 9 A directed acyclic monotone outerplanar graph that does not admit a 3-page book embedding. Note that the graph is 4-page book embeddable; the linear order of one such 4-page embedding is given by the vertex labels, while the page assignment by the coloring of the edges.

5 Open problems

The most natural open question raised from our work is to close the gap of Theorem 2. Towards another improvement on the page number of general directed acyclic outerplanar graphs, one needs either to introduce improvements for each of the remaining steps in the approach by Jungeblut, Merker and Ueckerdt [15] that supports Theorem 1 or to come up with a new, novel approach. We deem both problems intriguing.

References

1 Michael A. Bekos, Martin Gronemann, and Chrysanthi N. Raftopoulou. Two-page book embeddings of 4-planar graphs. *Algorithmica*, 75(1):158–185, 2016. doi:10.1007/s00453-015-0016-8.

2 Michael A. Bekos, Michael Kaufmann, Fabian Klute, Sergey Pupyrev, Chrysanthi N. Raftopoulou, and Torsten Ueckerdt. Four pages are indeed necessary for planar graphs. *J. Comput. Geom.*, 11(1):332–353, 2020. doi:10.20382/JOCG.V11I1A12.

3 Michael A. Bekos, Michael Kaufmann, and Christian Zielke. The book embedding problem from a SAT-solving perspective. In Emilio Di Giacomo and Anna Lubiw, editors, *Graph Drawing and Network Visualization*, volume 9411 of *LNCS*, pages 125–138. Springer, 2015. doi:10.1007/978-3-319-27261-0_11.

4 Frank Bernhart and Paul C. Kainen. The book thickness of a graph. *J. Comb. Theory, Ser. B*, 27(3):320–331, 1979. doi:10.1016/0095-8956(79)90021-2.

5 Carla Binucci, Giordano Da Lozzo, Emilio Di Giacomo, Walter Didimo, Tamara Mchedlidze, and Maurizio Patrignani. Upward book embeddability of st-graphs: Complexity and algorithms. *Algorithmica*, 85(12):3521–3571, 2023. doi:10.1007/S00453-023-01142-Y.

6 Jonathan F. Buss and Peter W. Shor. On the pagenumber of planar graphs. In Richard A. DeMillo, editor, *ACM Symposium on Theory of Computing*, pages 98–100. ACM, 1984. doi:10.1145/800057.808670.

7 Fan R. K. Chung, Frank Thomson Leighton, and Arnold L. Rosenberg. Embedding graphs in books: A layout problem with applications to VLSI design. *SIAM Journal on Algebraic and Discrete Methods*, 8(1):33–58, 1987. arXiv:http://dl.acm.org/citation.cfm?id=21936.25439.

8 James Davies. Improved bounds for colouring circle graphs. *Proceedings of the American Mathematical Society*, 2022. doi:10.1090/proc/16044.

9 Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, and Stephen K. Wismath. Book embeddability of series-parallel digraphs. *Algorithmica*, 45(4):531–547, 2006. doi:10.1007/S00453-005-1185-7.

10 Vida Dujmović and David R. Wood. On linear layouts of graphs. *Discrete Mathematics & Theoretical Computer Science*, 6(2):339–358, 2004. doi:10.46298/DMTCS.317.

11 Hikoe Enomoto, Tomoki Nakamigawa, and Katsuhiro Ota. On the pagenumber of complete bipartite graphs. *J. Comb. Theory, Ser. B*, 71(1):111–120, 1997. doi:10.1006/jctb.1997.1773.

12 Lenwood S. Heath. Embedding planar graphs in seven pages. In *FOCS*, pages 74–83. IEEE Computer Society, 1984. doi:10.1109/SFCS.1984.715903.

13 Lenwood S. Heath, Sriram V. Pemmaraju, and Ann N. Trenk. Stack and queue layouts of directed acyclic graphs: Part I. *SIAM J. Comput.*, 28(4):1510–1539, 1999. doi:10.1137/S0097539795280287.

14 Paul Jungeblut, Laura Merker, and Torsten Ueckerdt. A sublinear bound on the page number of upward planar graphs. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *ACM-SIAM SODA*, pages 963–978. SIAM, 2022. doi:10.1137/1.9781611977073.42.

15 Paul Jungeblut, Laura Merker, and Torsten Ueckerdt. Directed acyclic outerplanar graphs have constant stack number. In *FOCS*, pages 1937–1952. IEEE, 2023. doi:10.1109/FOCS57990.2023.00118.

16 Tamara Mchedlidze and Antonios Symvonis. Crossing-free acyclic hamiltonian path completion for planar *st*-digraphs. In Yingfei Dong, Ding-Zhu Du, and Oscar H. Ibarra, editors, *ISAAC*, volume 5878 of *LNCS*, pages 882–891. Springer, 2009. doi:10.1007/978-3-642-10631-6_89.

17 Martin Nöllenburg and Sergey Pupyrev. On families of planar dags with constant stack number. In Michael A. Bekos and Markus Chimani, editors, *Graph Drawing*, volume 14465 of *LNCS*, pages 135–151. Springer, 2023. doi:10.1007/978-3-031-49272-3_10.

18 Richard Nowakowski and Andrew Parker. Ordered sets, pagenumbers and planarity. *Order*, 6:209–218, 1989. doi:10.1007/BF00563521.

19 T. Ollmann. On the book thicknesses of various graphs. In F. Hoffman, R.B. Levow, and R.S.D. Thomas, editors, *Southeastern Conference on Combinatorics, Graph Theory and Computing*, volume VIII of *Congressus Numerantium*, page 459, 1973.

20 Arnold L. Rosenberg. The diogenes approach to testable fault-tolerant arrays of processors. *IEEE Trans. Computers*, 32(10):902–910, 1983. doi:10.1109/TC.1983.1676134.

21 Arnold L. Rosenberg. Book embeddings and wafer-scale integration. In *Southeastern Conference on Combinatorics, Graph Theory and Computing*, volume 54 of *Congressus Numerantium*, pages 217–224, 1986.

22 Mihalis Yannakakis. Four pages are necessary and sufficient for planar graphs (extended abstract). In Juris Hartmanis, editor, *ACM Symposium on Theory of Computing*, pages 104–108. ACM, 1986. doi:10.1145/12130.12141.

23 Mihalis Yannakakis. Planar graphs that need four pages. *J. Comb. Theory, Ser. B*, 145:241–263, 2020. doi:10.1016/j.jctb.2020.05.008.

The Bend Number of Cocomparability Graphs

Todor Antić
Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Vit Jelínek
Computer Science Institute, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Martin Pergel
Department of Software and Computer Science Education, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Felix Schröder
Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Peter Stumpf
Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic
Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague, Czech Republic

Pavel Valtr
Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Abstract

We introduce a new complexity measure for cocomparability graphs of posets or in other words, intersection graphs of piecewise linear functions, the bend number. We prove that cocomparability graphs of bounded bend number are not too plentiful and give two hierarchies of classes of cocomparability graphs, depending on whether the piecewise linear functions are restricted to slopes of ± 1 (diagonal case) or not (general case). These hierarchies give a gradation between permutation graphs and cocomparability graphs.

2012 ACM Subject Classification Mathematics of computing → Graphs and surfaces

Keywords and phrases Intersection Graphs, Bend Number, Piecewise Linear Functions, Graph Class Hierarchy, Cocomparability Graphs, Permutation Graphs, Poset Dimension

Digital Object Identifier 10.4230/LIPIcs.GD.2025.10

Funding *Todor Antić*: supported by grant no. 23-04949X of the Czech Science Foundation (GAČR) and by SVV–2023–260699.
Vit Jelínek: supported by grant no. 23-04949X of the Czech Science Foundation (GAČR).
Felix Schröder: supported by grant no. 23-04949X of the Czech Science Foundation (GAČR).
Peter Stumpf: supported by grant no. 23-04949X of the Czech Science Foundation (GAČR).
Pavel Valtr: supported by grant no. 23-04949X of the Czech Science Foundation (GAČR).

Acknowledgements We would like to thank Ida Kantor, Jelena Glišić and Maria Saumell for helpful discussions about the problems considered in this paper. We would also like to thank anonymous reviewers for their comments and suggestions, which helped improve the quality of this work.



© Todor Antić, Vit Jelínek, Martin Pergel, Felix Schröder, Peter Stumpf, and Pavel Valtr;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

The *cocomparability graph* of a poset $(P, \preceq)$ is a graph whose vertices are the elements of P, with two vertices connected by an edge if and only if they are incomparable in $(P, \preceq)$. In 1983, Golumbic, Rotem and Urrutia [16] showed that cocomparability graphs can be also equivalently characterised as intersection graphs of x-monotone curves that project orthogonally to the interval $[0, 1]$ on the x-axis; more explicitly, $G = (V, E)$ is a co-comparability graph if and only if, for an interval I, we can associate to each vertex v a continuous function $f_v \colon I \to \mathbb{R}$ so that two vertices v, w are adjacent if and only if $f_v(x) = f_w(x)$ for some $x \in I$. We then call such a collection of functions $(f_v;\ v \in V)$ a *cocomparability representation* of G.

In practice, it is often convenient, and sometimes necessary, to consider representations that admit a simple combinatorial description, for instance, to restrict the functions f_v to be piecewise linear with not too many bends.

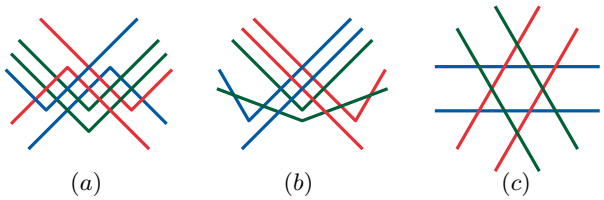


Figure 1 Cocomparability representations of the octahedral graph $K_{2,2,2}$ in the diagonal setting (a) and the general setting (b), (c).

In this paper, we initiate a systematic study of the classes of graphs admitting a co-comparability representation by piecewise linear curves with at most k bends per curve. In fact, we separately consider two natural settings: the *diagonal setting*, where the curves are made of alternating segments of slopes $+1$ and -1, and the *general setting* where arbitrary slopes are allowed (see Figure 1). Within each of the two settings, we distinguish *bounded* representations, in which the common domain I is a bounded interval $[a, b]$, *grounded* representations, in which I is of the form $[a, \infty)$, and *unbounded* representations, in which I is the whole $\mathbb{R}$. Moreover, in the diagonal setting, we may further distinguish representations in which all the curves have precisely k bends (so-called *strict representations*), from those whose curves have at most k bends (the *relaxed representations*). And finally, in the strict diagonal setting, we may impose the restriction that the leftmost segments of all the curves have the same slope, say $+1$ (so-called *synchronous representation*), or we may allow the leftmost slopes to differ (*asynchronous representation*).

The aim of this paper is to clarify the relative expressive power of the various forms of k-bend cocomparability representations. In particular, we prove that in both the diagonal and the general setting, the class of (diagonal/general) k-bend cocomparability graphs is a strict subclass of the class of (diagonal/general) $(k + 1)$-bend cocomparability graphs.

Previous results

Intersection graphs of (not necessarily x-monotone) piecewise linear curves with bounded bend number have been extensively studied. Asinowski, Cohen, Golumbic, Limouzy, Lipshteyn and Stern [3, 4] have introduced the graph class B_k-VPG, which are the intersection graphs of piecewise-linear curves whose segments are horizontal and vertical and which have at most k bends. They showed that B_1-VPG is a strict superset of B_0-VPG, and conjectured

that B_k-VPG is a strict superset of B_{k-1}-VPG for each k. The conjecture has been subsequently proven by Chaplick, Jelínek, Kratochvíl and Vyskočil [8]. Up to a 45-degree rotation, every diagonal cocomparability representation with k bends per curve is also a B_k-VPG-representation. However, the arguments used by [8] to distinguish B_k-VPG from B_{k-1}-VPG do not yield cocomparability graphs, so they are not helpful for our purposes.

A notable subclass of B_1-VPG are the intersection graphs of L-shapes, where an L-shape is a union of a horizontal and a vertical segment in which the bottom endpoint of the vertical segment coincides with the left endpoint of the horizontal one. This class of graphs are known as *L-graphs*. It is known that all planar graphs are L-graphs [17], while every L-graph is also an intersection graph of straight line segments [21]. Furthermore, complements of planar graphs are known to belong to B_{17}-VPG [14].

Cocomparability graphs have been considered in the context of bend-bounded representations as well. In particular, various authors studied the relationship between the necessary number of bends in various representations and the dimension of the poset represented by the cocomparability graph. Already in the seminal paper by Golumbic, Rotem and Urrutia [16], it has been pointed out that a cocomparability graph of poset dimension $d \geq 2$ can be represented via piecewise linear curves with at most $d-2$ bends and arbitrary slopes, as in our general setting. Later, Cohen, Golumbic, Trotter and Wang [9] showed that cocomparability graphs of posets of dimension $d \geq 2$ are in B_{d-1}-VPG; however, their argument does not yield x-monotone curves (even after rotation), and therefore not a bend-bounded cocomparability representation of the type we consider in this paper. More refined results on the relationship of poset dimension and B_k-VPG membership were obtained by Chakraborty, Das, Mukherjee and Sahoo [6], again with no attempt to obtain cocomparability representations.

To our knowledge, the first work to explicitly consider bend-bounded diagonal cocomparability representations is by Nesterenko [22]. His main motivation was the class of grounded-unit-L graphs, which are intersection graphs of L-shapes whose top endpoints belong to a common horizontal line and all the horizontal segments of the L-shapes have length 1. This subclass of cocomparability graphs was previously considered by Chakraborty, Gajjar and Rusu [7]. In our terminology, Nesterenko showed grounded-unit-L graphs are a strict superclass of the graphs with asynchronous unbounded 1-bend cocomparability representation, while being incomparable to those with bounded 1-bend cocomparability representation, and being a strict subclass of those with a synchronous 2-bend representation.

Nesterenko [22] also pointed out that cocomparability graphs of posets with dimension d have a synchronous representation with $d-1$ bends, slightly strengthening the result from [9]; on the other hand, he pointed out that cocomparability graphs with 1-bend representations may represent posets of arbitrary dimension. An open problem of Nesterenko [22] was to clarify the inclusions between the various graph classes defined by bend-bounded diagonal cocomparability representations. This problem serves as a starting point for our paper.

2 Notation

Throughout this paper, we work with x-monotone piecewise linear curves with a finite number of bends. For $k \geq 1$, a *k-bend curve* is a concatenation of a sequence $s_1, s_2, \ldots, s_{k+1}$ of nonvertical segments and nonvertical rays, where the rightmost point of s_i coincides with the leftmost point of s_{i+1}. In particular, $s_2, \ldots, s_k$ are necessarily segments, while each of s_1 and s_{k+1} is either a segment or a ray. A 0-bend curve is a nonvertical line. We call s_i the *i-th segment* of the curve (even when it is in fact a ray), and the common endpoint of s_i and s_{i+1} is the *i-th bend* of the curve. We always assume that s_i and s_{i+1} have different slopes,

so the decomposition of a curve into segments is unique. A k-bend curve is necessarily the graph of a continuous function $f\colon I \to \mathbb{R}$ defined on an interval I. Where there is no risk of confusion, we identify the function f with its graph, and refer to k-bend curves as 'functions'.

A *diagonal curve* is a k-bend curve whose every segment has a slope equal to $+1$ or -1. The segments with slope $+1$ are the *increasing segments*, while the others are *decreasing*. If a segment s_i is increasing, then s_{i+1} (if it exists) is decreasing and their common point is then a *peak* of the curve. Bends that are not peaks are the *valleys* of the curve. A *k-bend representation* of a graph $G = (V, E)$ associates to each vertex $v \in V$ a k'-bend curve f_v for some $k' \leq k$, so that all the curves $\{f_v;\ v \in V\}$ have the same domain I, and in such a way, that two vertices v, w are adjacent in G if and only if the two curves f_v and f_w intersect. A k-bend representation is *pure* if any two of its curves share at most finitely many points, no two curves share an endpoint, no three curves share a common point, and no bend of a curve belongs to another curve. We will henceforth assume that all considered representations are pure.

Even in the diagonal setting, by choosing independently various combinations of bounded/-grounded/unbounded, strict/non-strict, synchronous/non-synchronous restrictions, we might in theory get twelve representation models. The following proposition shows, however, that many of these models have the same expressive power.

▶ **Proposition 1.**

1. *Any unbounded representation Γ can be transformed into a grounded representation Γ' of the same graph, and any grounded representation Γ' can be transformed into a bounded representation Γ'' of the same graph, in such a way that the number of bends and the slopes of the segments of each curve are preserved.*
2. *If a graph G has a grounded or bounded k-bend representation Γ, then it also has a strict k-bend representation Γ' on the same domain. Moreover, if Γ is diagonal, then so is Γ'.*
3. *If a graph G has a synchronous diagonal k-bend representation Γ, then it also has a strict synchronous diagonal k-bend representation on the same domain.*
4. *If a graph G has a strict synchronous diagonal k-bend representation which is bounded or grounded, then it also has an unbounded strict synchronous diagonal k-bend representation.*

Proof. (1) For an unbounded representation Γ, choose a constant a small enough so that any bend of Γ and any intersection of two curves from Γ appears to the right of the line $x = a$. Then obtain Γ' by restricting the functions in Γ to the interval $[a, +\infty)$. In a similar way, choosing b large enough and restricting to the domain $[a, b]$, we obtain Γ''.

(2) We can add any number of extra bends and segments in a small neighborhood of the leftmost point of a curve without affecting the intersection graph.

(3) If Γ is bounded, we can add more bends near the right endpoint of any curve with fewer than k bends, until we get a strict representation. If Γ is grounded or unbounded, we proceed as follows: choose a $b \in \mathbb{R}$ large enough so that all the crossing points and bends of Γ are to the right of the line $x = b$. Then add to each curve of Γ a suitable number of bends in a small neighborhood of the point where the curve crosses the line $x = b$, so that all curves have exactly k bends. Since the leftmost segments of all the curves in Γ are parallel, the same is true for rightmost segments after the above modification. In particular, the modification does not create any new intersections between curves.

(4) We can turn Γ into an unbounded representation by simply extending the leftmost and the rightmost segment of each curve into infinity. Since Γ is strict and synchronous, this cannot create any new intersections between curves. ◀

We are now ready to formally define the graph classes of interest. Let us begin with the diagonal case. For each $k \in \mathbb{N}_0$, by Proposition 1 we should consider five classes:

- $\mathfrak{D}_k^{\nearrow}$ is the class of graphs that have a synchronous diagonal k-bend representation. By Parts 3 and 4 of Proposition 1, any such representation can be transformed to a representation that is additionally strict and unbounded. Thus, a $\mathfrak{D}_k^{\nearrow}$*-representation* of a graph G is a synchronous strict unbounded diagonal k-bend representation of G.
- $\mathfrak{D}_k^{=}$ is the class of graphs that have a strict unbounded diagonal k-bend representation. Such a representation is then called a $\mathfrak{D}_k^{=}$*-representation*.
- $\mathfrak{D}_k^{\leq}$ is the class of graphs that have an unbounded diagonal k-bend representation, which we call a $\mathfrak{D}_k^{\leq}$*-representation*.
- $\mathfrak{D}_k^{\vdash}$ is the class of graphs that have a grounded diagonal k-bend representation; by Part 2 of Proposition 1, we may additionally assume that the representation is strict. A $\mathfrak{D}_k^{\vdash}$*-representation* is thus a strict grounded diagonal k-bend representation.
- $\mathfrak{D}_k^{\mathsf{H}}$ is the class of graphs that have a bounded diagonal k-bend representation, and by Part 2 of Proposition 1, we may again assume that the representation is strict. A $\mathfrak{D}_k^{\mathsf{H}}$*-representation* is then a strict bounded diagonal k-bend representation.

For general (as opposed to diagonal) k-bend representations, the distinction between synchronous and non-synchronous representations does not apply, and the distinction between strict and non-strict representations is mostly irrelevant, since one can always add new bends by replacing a single bend by two or more bends close to each other. Thus, we only need to consider three types of general representations, for each $k \in \mathbb{N}_0$.

- $\mathfrak{G}_k^{-}$ is the class of graphs that have an unbounded k-bend representation.
- $\mathfrak{G}_k^{\vdash}$ is the class of graphs that have a grounded k-bend representation.
- $\mathfrak{G}_k^{\mathsf{H}}$ is the class of graphs that have a bounded k-bend representation.

Our results

Our goal is to characterise the inclusions among the graph classes introduced above. For the diagonal case, we succeeded in obtaining a complete characterisation of all the inclusions.

▶ **Theorem 2** (Main result – diagonal case)**.**

- *For any odd* $k \geq 1$*, we have* $\mathfrak{D}_{k-1}^{\mathsf{H}} \subsetneq \mathfrak{D}_k^{\nearrow} \subsetneq \mathfrak{D}_k^{=} = \mathfrak{D}_k^{\leq} = \mathfrak{D}_k^{\vdash} \subsetneq \mathfrak{D}_k^{\mathsf{H}}$.
- *For any even* $k \geq 2$*, we have* $\mathfrak{D}_{k-1}^{\mathsf{H}} \subsetneq \mathfrak{D}_k^{\nearrow} \subsetneq \mathfrak{D}_k^{=} \subsetneq \mathfrak{D}_k^{\leq} \subsetneq \mathfrak{D}_k^{\vdash} \subsetneq \mathfrak{D}_k^{\mathsf{H}}$.

The proof of this result appears in Section 3. We remark that for the case $k = 0$, one easily observes the inclusions $\mathfrak{D}_0^{\nearrow} \subsetneq \mathfrak{D}_0^{=} = \mathfrak{D}_0^{\leq} \subsetneq \mathfrak{D}_0^{\vdash} \subsetneq \mathfrak{D}_0^{\mathsf{H}}$.

For the general case, we are able to show that increasing the number of bends strictly increases the expressive power of the model considered.

▶ **Theorem 3** (Main result – general case)**.** *For any* $k \geq 0$, $\mathfrak{G}_k^{-} \subseteq \mathfrak{G}_k^{\vdash} = \mathfrak{G}_k^{\mathsf{H}} \subsetneq \mathfrak{G}_{k+1}^{-}$.

We prove this in Section 4. We do not know whether the inclusion $\mathfrak{G}_k^{-} \subseteq \mathfrak{G}_k^{\mathsf{H}}$ is strict (except for several small values of k), but we conjecture that it is.

Before we turn to the proofs of our results, we introduce several graph constructions that are used throughout the paper. For two disjoint graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the *complete join* (or just *join*) of G_1 and G_2 is the graph obtained from the disjoint union of G_1 and G_2 by adding all the edges between V_1 and V_2. For a graph $G = (V, E)$, a set of vertices $A \subseteq V$ and an integer $t \geq 1$, we say that a set $B \subseteq V \setminus A$ is a *t-extension of A* if the following holds:

- every vertex from B is adjacent to exactly t vertices from A,

- for each set $S \subseteq A$ containing t vertices, there is a unique vertex $x_S \in B$ which is adjacent to the vertices in S (and therefore to no vertex in $A \setminus S$), and
- any two vertices in B are adjacent.

For a pair of graphs G and H, we let $G + H$ denote the disjoint union of G and H.

3 Diagonal setting

In this section, we focus on the relationships between the classes $\mathfrak{D}_k^{\nearrow}$, $\mathfrak{D}_k^{=}$, $\mathfrak{D}_k^{\leq}$, $\mathfrak{D}_k^{\vdash}$, and $\mathfrak{D}_k^{\mathsf{H}}$. We begin by noticing that these classes form a hierarchy.

▶ **Theorem 4.** *For any $k \geq 0$, $\mathfrak{D}_k^{\nearrow} \subseteq \mathfrak{D}_k^{=} \subseteq \mathfrak{D}_k^{\leq} \subseteq \mathfrak{D}_k^{\vdash} \subseteq \mathfrak{D}_k^{\mathsf{H}} \subseteq \mathfrak{D}_{k+1}^{\nearrow}$.*

Proof. Any $\mathfrak{D}_k^{\nearrow}$-representation is also a $\mathfrak{D}_k^{=}$-representation, and any $\mathfrak{D}_k^{=}$-representation is also a $\mathfrak{D}_k^{\leq}$-representation, showing that $\mathfrak{D}_k^{\nearrow} \subseteq \mathfrak{D}_k^{=} \subseteq \mathfrak{D}_k^{\leq}$. The inclusions $\mathfrak{D}_k^{\leq} \subseteq \mathfrak{D}_k^{\vdash} \subseteq \mathfrak{D}_k^{\mathsf{H}}$ follow from Part 1 of Proposition 1. It remains to prove that $\mathfrak{D}_k^{\mathsf{H}} \subseteq \mathfrak{D}_{k+1}^{\nearrow}$.

If we are given a $\mathfrak{D}_k^{\mathsf{H}}$-representation Γ of G with domain $[a, b]$ for some $a, b \in \mathbb{R}$, we define a $\mathfrak{D}_{k+1}^{\nearrow}$-representation Γ' as follows. Let f be a function in Γ. If the leftmost segment of f has slope -1, we add to f a new leftmost segment with slope $+1$ with domain $(-\infty, a]$, and we extend the rightmost segment of f (which has slope $(-1)^{k+1}$) to $+\infty$. On the other hand, if the leftmost segment of f has slope $+1$, we extend it to $-\infty$ and add an additional rightmost segment to f with slope $(-1)^{k+1}$ and domain $[b, +\infty)$. This new representation is clearly a $\mathfrak{D}_{k+1}^{\nearrow}$-representation since we added one new bend to each function, and it is still a representation of G since we did not introduce any new crossings. This shows that $\mathfrak{D}_k^{\mathsf{H}} \subseteq \mathfrak{D}_{k+1}^{\nearrow}$, finishing the proof. ◀

The remainder of this section is dedicated to determining which of these inclusions are strict. To do so systematically, we differentiate between odd k and even k.

3.1 Odd number of bends

▶ **Theorem 5.** *For k odd, $\mathfrak{D}_k^{=} = \mathfrak{D}_k^{\leq} = \mathfrak{D}_k^{\vdash}$.*

Proof. By Theorem 4, we have $\mathfrak{D}_k^{=} \subseteq \mathfrak{D}_k^{\leq} \subseteq \mathfrak{D}_k^{\vdash}$. Now, let G be a graph in $\mathfrak{D}_k^{\vdash}$ and let Γ be a $\mathfrak{D}_k^{\vdash}$-representation of G. Recall that we can assume that Γ is strict. We construct a $\mathfrak{D}_k^{=}$-representation of G from Γ by extending the leftmost segment of each function to $-\infty$. This new representation Γ' is clearly a $\mathfrak{D}_k^{=}$-representation, but we need to show that it is a representation of G. Clearly, if two functions cross in Γ, they still cross in Γ'. Now, let f, g be two functions that do not cross in Γ, say $f \leq g$. We need to show that f, g still do not cross in Γ'. This is clearly true if the leftmost segment of f has greater or equal slope than g, so we may assume this slope is -1 while the leftmost segment of g has slope $+1$. Since k is odd and the representation is strict, this means that the rightmost segments of f and g have slopes $+1$ and -1 respectively, a contradiction to $f \leq g$. ◀

Our next two results deal with constructions that fail to be representable within a restricted class of curves. They both depend on a geometric counting argument which we now introduce. For a set of points P and an unbounded k-bend curve f, we say that f *cuts off* a subset $A \subseteq P$ from P, if the points from A are all strictly below the curve f, while the points of $P \setminus A$ are strictly above it.

▶ **Lemma 6.** *For every $m \in \mathbb{N}$ and $t \in \mathbb{N}_0$ there is a constant $c = c(m, t)$ such that every nonempty finite set of points P in the plane has at most $c|P|^t$ subsets of size at most m that can be cut off from P by an unbounded diagonal curve with at most t peaks.*

Proof. Fix m and fix a point set P of size $N \geq 1$. We say that a set $A \subseteq P$ is *small* if $|A| \leq m$.

Let us proceed by induction on t. For $t = 0$, the curves under consideration are lines of slopes $+1$ or -1 and 1-bend diagonal curves with a single valley. There are at most $m+1$ small subsets of P that can be cut off by a line of slope $+1$, and likewise at most $m+1$ cut off by lines of slope -1. If A is a small set cut off by a diagonal 1-bend curve with a valley at a point X, then A is the union of two sets A^+ and A^- where A^+ is cut off by the line of slope $+1$ passing through X and A^- by a line of slope -1 through X. As there are at most $(m+1)$ choices for A^+ and for A^-, there are at most $(m+1)^2$ such sets A.

Suppose now that $t = 1$ and consider the curves that have one peak Y and no valleys. Any small set cut off by a curve of this form can be identified in two steps:

1. Choose a line ℓ of slope $+1$.
2. Choose a point $Y \in \ell$ so that the 1-bend diagonal curve with a single peak in Y cuts off a small subset of P.

In the first step, we can only make $N+1$ combinatorially distinct choices, since we only need to know which points of P are below ℓ. In the second step, there are at most $m+1$ distinct small sets that can be cut off once ℓ is fixed. Thus, at most $(m+1)(N+1)$ small sets can be cut off by curves with a single peak and no valleys.

For a curve f with a single peak Y and up to two valleys X_1, X_2 with X_1 left of X_2, the set of points cut off by f is the union of the points cut off by the decreasing line through X_1, the points cut off by the 1-bend curve with peak at Y, and the points cut off by the increasing line through X_2. The number of small sets obtainable in this way is at most $(m+1)^3(N+1)$.

For $t \geq 2$, consider a curve f with t peaks, and let X be the valley lying between the first and second peak of f. Consider the unbounded diagonal curve f_1 whose bends are exactly the bends of f strictly to the left of X, and the unbounded diagonal curve f_2 whose bends are the bends of f strictly to the right of X; in particular, f_1 and f_2 both pass through X, f_1 has 1 peak and f_2 has $t-1$ peaks. The set of points cut off by f is the union of the set cut off by f_1 and the set cut off by f_2. By induction, there are at most $c(m,1)c(m,t-1)N^t$ distinct small sets cut off by a curve with t peaks. ◀

In our applications of Lemma 6, we will assume that m and t are constants, while $|P|$ will grow to $+\infty$. We then write the bound $c(m,t)|P|^t$ from the lemma simply as $\mathcal{O}_{m,t}(|P|^t)$.

▶ **Theorem 7.** *For every k odd, there is a graph $G \in \mathfrak{D}_k^{\nearrow} \setminus \mathfrak{D}_{k-1}^{\mathsf{H}}$.*

Proof. We have $\mathfrak{D}_0^{\mathsf{H}} \subsetneq \mathfrak{D}_1^{\nearrow}$, since $\mathfrak{D}_0^{\mathsf{H}}$ only contains bipartite permutation graphs, while $\mathfrak{D}_1^{\nearrow}$ is known to be the class of all permutation graphs [22]. Now assume $k \geq 3$, and let $t = \frac{k+1}{2}$.

The separation is witnessed by the graph $G(n)$ consisting of a clique A of size n with a t-extension B. We easily observe that for any n, $G(n)$ has a $\mathfrak{D}_k^{\nearrow}$-representation, see Figure 2.

We will show that for n large enough, $G(n)$ has no $\mathfrak{D}_{k-1}^{\mathsf{H}}$-representation. For contradiction, consider a $\mathfrak{D}_{k-1}^{\mathsf{H}}$-representation Γ of $G(n)$. For any $a \in A$ and $b \in B$ nonadjacent vertices, the curve f_a is either completely above or completely below f_b. Without loss of generality, suppose that f_a is above f_b, otherwise we turn Γ upside down. It follows that any other vertex $b' \in B$ nonadjacent to a has its representation below f_a, because f_b and $f_{b'}$ must cross. We conclude that for any pair of nonadjacent vertices $a \in A$ and $b \in B$, the curve f_a is above f_b.

Let P be the set of all the endpoints and bends of all the curves of Γ representing vertices in A. In particular, $|P| = n(k+2)$, since we may assume that Γ is strict. Every curve representing a vertex from b crosses exactly t curves representing vertices in A, and hence it

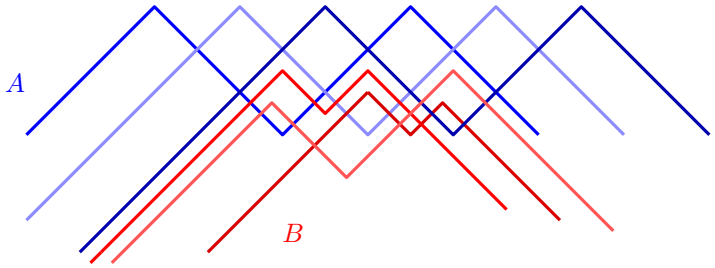


Figure 2 A $\mathfrak{D}_3^{\nearrow}$-representation of the graph $G(3)$ from the proof of Theorem 7. For readability, colors of the sets A, B vary slightly.

cuts off at most $t(k+2)$ points of P. Moreover, two curves representing distinct vertices from B must cut off distinct subsets of P. Thus, at least $\binom{n}{t}$ distinct subsets of P of size at most $(k+2)t$ are cut off in this way. However, the curves of Γ have at most $t-1$ peaks, and therefore can cut off at most $\mathcal{O}_{k,t}(n^{t-1})$ distinct subsets of P of size at most $t(k+2)$ by Lemma 6. For n large enough, this yields a contradiction. ◀

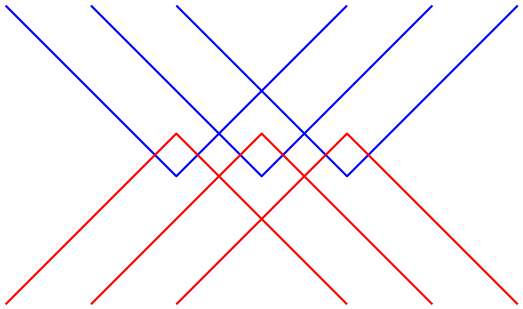


Figure 3 A $\mathfrak{D}_1^{=}$-representation of the graph $\overline{C_6}$, which is not a permutation graph and therefore has no $\mathfrak{D}_1^{\nearrow}$-representation.

▶ **Theorem 8.** *For every k odd, there is a graph $G \in \mathfrak{D}_k^{=} \setminus \mathfrak{D}_k^{\nearrow}$.*

Proof. For $k = 1$ the theorem holds, since $\mathfrak{D}_1^{\nearrow}$ is the class of permutation graphs, while $\mathfrak{D}_1^{=}$ contains the graph $\overline{C_6}$ (the complement of the 6-cycle) which is not a permutation graph (see Figure 3). Suppose that $k \geq 3$ and let $t = \frac{k+1}{2}$.

Let $G(n)$ be the graph consisting of a clique B of size n and two disjoint t-extensions A and C of B. There are no edges between A and C. We easily observe that G has a $\mathfrak{D}_k^{=}$-representation, see Figure 4.

We will show that for n large enough, $G(n)$ has no $\mathfrak{D}_k^{\nearrow}$-representation.

Suppose for contradiction that $G(n)$ has a $\mathfrak{D}_k^{\nearrow}$-representation Γ. In Γ, for any choice of pairwise nonadjacent vertices $a \in A$, $b \in B$ and $c \in C$, the curve f_b must be between f_a and f_c. Suppose without loss of generality that f_b is below f_a and above f_c. Since all vertices of A are adjacent to a and all vertices of C are adjacent to c, it follows that for any choice of $a' \in A$, $b' \in B$ and $c' \in C$ pairwise nonadjacent, $f_{b'}$ is below $f_{a'}$ and above $f_{c'}$.

Each curve in Γ has t peaks and $t-1$ valleys. Let us turn Γ upside down, and then transform it into a bounded representation by limiting its domain to a bounded interval covering all intersection points and bends. Call the modified representation Γ'. We now repeat the argument of the proof of Theorem 7: let P be the set of bends and endpoints of

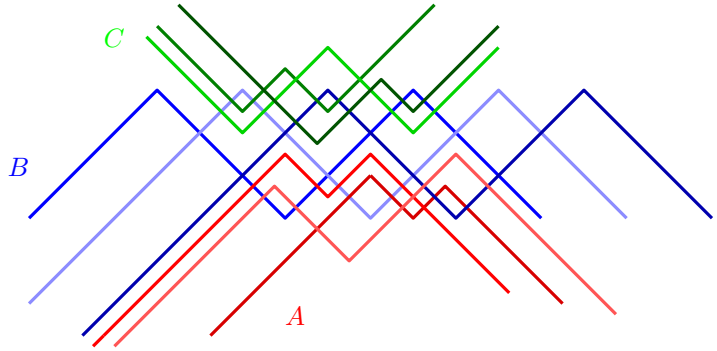


Figure 4 A $\mathfrak{D}_3^{=}$-representation of the graph $G(3)$ from the proof of Theorem 8. For readability, the colors of the sets A, B, C vary slightly.

the curves representing B in Γ'. Each curve representing a vertex $a \in A$ in Γ' has at most $t-1$ peaks, and cuts off from P a subset of size at most $t(k+2)$. By Lemma 6, there are at most $\mathcal{O}_k(n^{t-1})$ such subsets of P, whereas A has $\binom{n}{t}$ vertices, and each must correspond to a different set being cut off. For n large enough, this yields a contradiction. ◀

We complete the case of k odd by a result for odd and even numbers of bends.

▶ **Theorem 9.** *For any k and for any graph $G \in \mathfrak{D}_k^{=} \setminus \mathfrak{D}_k^{\nearrow}$ we have $G + G \in \mathfrak{D}_k^{\mathsf{H}} \setminus \mathfrak{D}_k^{\vdash}$.*

Proof. Fix $G \in \mathfrak{D}_k^{=} \setminus \mathfrak{D}_k^{\nearrow}$. It follows that G is also in $\mathfrak{D}_k^{\mathsf{H}}$, and we easily see that $\mathfrak{D}_k^{\mathsf{H}}$ is closed under disjoint unions, hence $G + G \in \mathfrak{D}_k^{\mathsf{H}}$. For contradiction, assume that $G + G$ has a $\mathfrak{D}_k^{\vdash}$-representation Γ. If every component of G is in $\mathfrak{D}_k^{\nearrow}$, then G itself is in $\mathfrak{D}_k^{\nearrow}$, a contradiction. Hence, G has a connected component C not belonging to $\mathfrak{D}_k^{\nearrow}$.

Let C_1 and C_2 be two components isomorphic to C inside $G+G$. In Γ, the representations of C_1 and C_2 must include curves with increasing unbounded segments as well as those with decreasing unbounded segments, otherwise C would have a $\mathfrak{D}_k^{\nearrow}$-representation. Since each copy of C is connected, the left endpoints of its curves must appear consecutively on the grounding line of Γ. It follows that in Γ, a curve from the representation of C_1 crosses a curve from the representation of C_2, which is impossible. ◀

3.2 Even number of bends

Let $k = 2t$ be the number of bends. We will start by defining a graph $H_t(n)$ which, as we will prove, is in $\mathfrak{D}_k^{\nearrow} \setminus \mathfrak{D}_{k-1}^{\mathsf{H}}$ if n is large enough. This construction will further be used as a building block in the subsequent constructions of graphs in $\mathfrak{D}_k^{=} \setminus \mathfrak{D}_k^{\nearrow}$, $\mathfrak{D}_k^{\leq} \setminus \mathfrak{D}_k^{=}$, $\mathfrak{D}_k^{\vdash} \setminus \mathfrak{D}_k^{\leq}$ and $\mathfrak{D}_k^{\mathsf{H}} \setminus \mathfrak{D}_k^{\vdash}$.

We define $H_t(n)$ to be the graph consisting of a base clique $C = [n]$ and four cliques A, B, D and E, where B and D are disjoint t-extensions of C, while A is a t-extension of B and E is a t-extension of D. There are no other edges in $H_t(n)$. We call C the *central clique* of $H_t(n)$.

It is straightforward to show that $H_t(n)$ belongs to $\mathfrak{D}_{2t}^{\nearrow}$. We leave the details of the argument for the full version of our paper and refer to Figure 5.

The main part of this section is an analysis of the possible k-bend representations of $H_t(n)$. The aim is to prove that if for every n, the graph $H_t(n)$ can be represented within a given class $\mathfrak{D}$, then it also has such a representation satisfying certain additional restrictions. This analysis will serve us for two purposes: firstly, it will allow us to show that $H_t(n)$

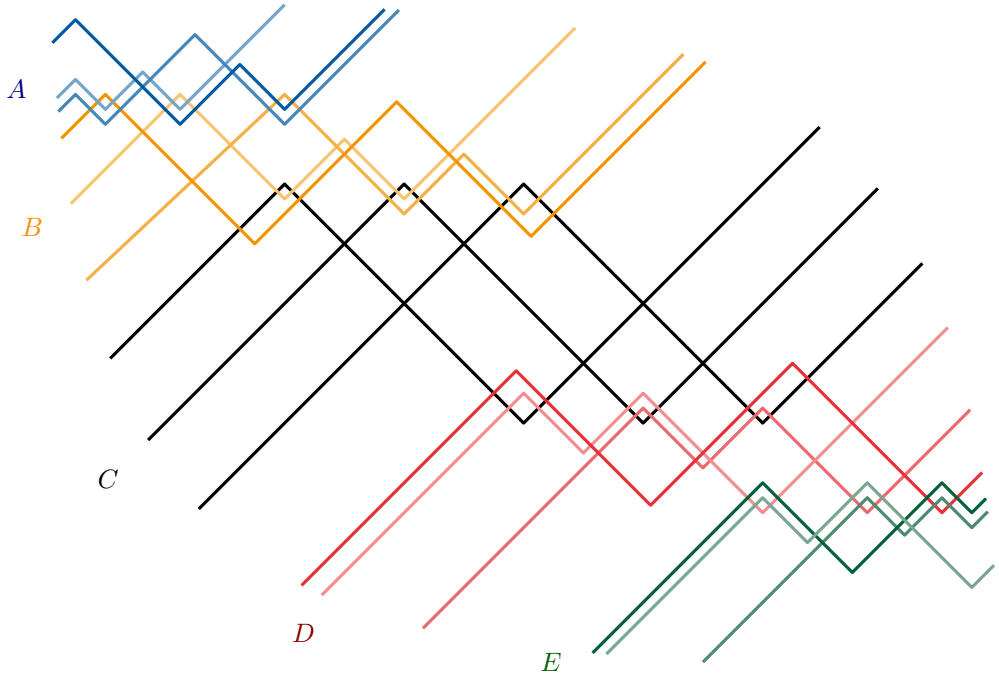


Figure 5 $\mathfrak{D}_4^{\nearrow}$-Representation of $H_2(3)$. For readability, colors of the sets A, B, D, E vary slightly.

has no $\mathfrak{D}_{2t-1}^{\mathsf{H}}$-representation, and secondly, it will impose constraints on the possible $\mathfrak{D}_{2t}^{\nearrow}$-representations, which will turn out useful when we use $H_t(n)$ as a building block for further constructions. To avoid duplicating our efforts, and highlight the generality of our arguments, we will now fix a number of bends $b \geq 1$ and a class $\mathfrak{D} \in \{\mathfrak{D}_b^{\nearrow}, \mathfrak{D}_b^{=}, \mathfrak{D}_b^{\vdash}, \mathfrak{D}_b^{\mathsf{H}}\}$, and study the properties of $\mathfrak{D}$-representations of $H_t(n)$ (we omit $\mathfrak{D}_b^{\leq}$, since it is convenient for us to work with $\mathfrak{D}$-representations that are strict).

Since we only want to investigate $H_t(n)$ for large enough n, our proof is making extensive use of pigeonhole and Ramsey-type arguments. More precisely, in order to obtain a $\mathfrak{D}$-representation of $H_t(n)$ with certain additional desirable properties, we typically start with a $\mathfrak{D}$-representation Γ of a larger graph $H_t(N)$ with central clique C. We then identify within C a subset C' of size n, and we delete from Γ all the vertices of $C \setminus C'$, as well as all the vertices in B and D adjacent to at least one deleted vertex in C, and all the vertices in A and E adjacent to at least one deleted vertex from B or D. The remaining vertices will form a representation Γ' of $H_t(n)$ with central clique C'. We will say that Γ' is the *restriction* of Γ generated by the set C'.

The above process will be used iteratively to impose increasingly stronger restrictions on the $\mathfrak{D}$-representations of $H_t(n)$. To avoid repetitive language, we will use the phrase "we can enforce property P" as a shorthand for the following claim: "For every $n \in \mathbb{N}$ there is an $N \in \mathbb{N}$ such that every $\mathfrak{D}$-representation of $H_t(N)$ can be restricted to a representation of $H_t(n)$ that satisfies property P."

Let us assume that for every n, the graph $H_t(n)$ has a $\mathfrak{D}$-representation. We let f_v denote the curve representing the vertex v in the $\mathfrak{D}$-representation of $H_t(n)$ being considered.

The first property we consider does not even need to be enforced by restrictions.

▷ Property 1. We may assume without loss of generality that for every $a \in A$, $c \in C$ and $e \in E$ we have $f_a > f_c > f_e$.

Proof. Since A, C and E are disjoint cliques, all functions in one of them lie above or below all functions of another. If C is not in the middle, then A is in the middle without loss of generality. However, this is impossible, because the functions of D are supposed to intersect functions of C and E while not intersecting any from A. Thus, C is in the middle. If A were below C and E above it, we could simply relabel the vertices, since $H_t(n)$ admits an automorphism that exchanges A with E and B with D. ◁

▷ Property 2. We may enforce that all the curves representing vertices in C have parallel leftmost segments.

Proof. Pick an arbitrary $\mathfrak{D}$-representation Γ of $H_t(2n-1)$, with central clique C. Clearly C contains a subset of n curves whose leftmost segments are parallel. By restricting to this subset, we enforce the property.

If C does not contain n upwards-starting curves, then it contains a set C'' of n downwards starting curves. We restrict Γ by this set C'', and turn the resulting representation upside down. To ensure that Property 1 is preserved, we again relabel vertices according to the automorphism that maps A to E and B to D. ◁

We may in fact assume without loss of generality that in the representations satisfying Property 2 the leftmost segments in the curves representing C are all increasing. This is automatic when $\mathfrak{D} = \mathfrak{D}_b^{\nearrow}$, and for $\mathfrak{D} \in \{\mathfrak{D}_{,}^{=}\mathfrak{D}_{,}^{\vdash}\mathfrak{D}^{\mathsf{H}}\}$, this can be achieved by turning the representation upside down and relabeling vertices to ensure Property 1 also holds. From now on, we will restrict ourselves to representations where the leftmost segments of the curves representing the central clique are all increasing.

Note that all curves in C have exactly b bends. Since they start upwards, they have exactly $\lceil \frac{b}{2} \rceil$ peaks and $\lfloor \frac{b}{2} \rfloor$ valleys. We will say that a point $(x, y) \in \mathbb{R}$ *lies above* another point (x', y'), if $y - y' > |x - x'|$. A set of points is *independent*, if no point of the set lies above any other. Note that since all functions in the diagonal representation are Lipschitz-continuous with Lipschitz constant 1, if a point on the curve f lies above a point on the curve g, then $f \geq g$ on the interval in between those two points.

From now on, we will scale our $\mathfrak{D}$-representations in such a way, that the x-coordinate every bend and every intersection point is in the interval $(0, 1)$. Furthermore, the domain of a bounded representation is taken to be $[0, 1]$ and the domain of a grounded representation is $[0, +\infty)$.

▷ Property 3. We may enforce that for any $c \in C$ and $d \in D$ we have $f_c(0) > f_d(0)$ and $f_c(1) > f_d(1)$.

Proof. Start with a representation of $H_t(n + 2t)$ satisfying the previously enforced properties. Choose $d \in D$ so that $f_d(0)$ is maximised. Note that if there is a $c \in C$ such that $f_c(0) < f_d(0)$, then f_d crosses f_c. This is because d is adjacent to a vertex $e \in E$, and $f_e < f_c$ by Property 1. Since d has t neighbors in C, there can be at most t vertices $c \in C$ for which $f_c(0) < f_d(0)$. By deleting these at most t vertices, we enforce that for every $c \in C$ and $d \in D$, $f_c(0)$ is greater than $f_d(0)$. By a similar argument, we also enforce that $f_c(1) > f_d(1)$, after deleting another set of at most t vertices. ◁

Note that Property 3 implies that for any $c \in C$ and $d \in D$, f_c and f_d intersect an even number of times. Moreover, if P and Q are two consecutive intersections of f_c and f_d so that f_d stays above f_c between P and Q, then f_d has at least one peak and f_c at least one valley between P and Q, and the leftmost such peak of f_d lies above the leftmost such valley of f_c.

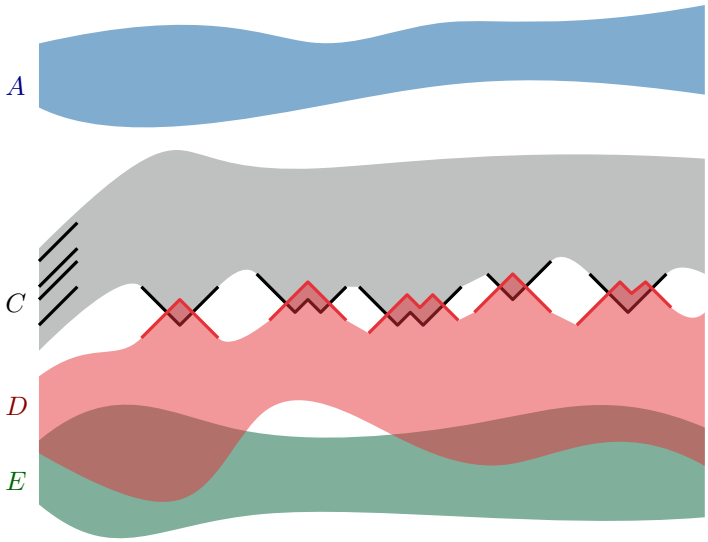


Figure 6 Visualization of properties 1-3.

From this point on, we will denote some of the valleys of the functions representing C as *active*. To determine which valleys are active, we first consider all of them as active, that is $\lfloor \frac{b}{2} \rfloor$ per function, then do $\lfloor \frac{b}{2} \rfloor$ steps: In step i, we consider all functions with exactly $\lfloor \frac{b}{2} \rfloor - i + 1$ active valleys. As long as an active valley v_1 of such a function f_1 representing $c_1 \in C$ lies above an active valley of such a function f_2 representing $c_2 \in C$, we delete c_2 from C and deactivate v_1. This procedure deletes at most half of the elements of C in each step, so at least $\frac{N}{2^{\lfloor \frac{b}{2} \rfloor}} \geq n$ elements remain. The procedure enforces the following two properties.

▷ Property 4. If a peak of a function f_d lies above a valley v of a function f_c, then v is active.

Proof. Assume for a contradiction that some function f_d obeys $f_d(x) > f_c(x)$, where (x, y) is a valley of f_c that is not active. Since all valleys were active before the procedure, there was a certain step in which (x, y) was deactivated. In this step another function $f_{c'}$ was deleted, whose valley (x', y') obeys $y - y' > |x - x'|$. Since f_d is Lipschitz continuous with Lipschitz constant 1, we get

$$f_d(x') \geq f_d(x) - |f_d(x) - f_d(x')| > f_c(x) - 1 \cdot |x - x'| = y - |x - x'| > y' = f_{c'}(x').$$

Since $f_d(0) < f_{c'}(0)$, by continuity f_d crosses $f_{c'}$ and should have therefore been removed when c' and all of its neighbors were removed, a contradiction. ◁

▷ Property 5. For every $i \in [\lfloor \frac{b}{2} \rfloor]$, the active valleys of all functions representing C with exactly i active valleys are independent.

Proof. This is true after step $\lfloor \frac{b}{2} \rfloor - i + 1$ because it would not have ended otherwise. It stays true as the functions with i active valleys are not changed in any of the further steps. ◁

▷ Property 6. We may enforce that all functions representing C have m active valleys, for some $m \in \{0, 1, \ldots, \lfloor \frac{b}{2} \rfloor\}$.

Proof. Let $0 \leq m \leq \lfloor \frac{b}{2} \rfloor$ be a number that maximizes the number of functions of C with exactly m active valleys. At least $\frac{N}{\lfloor \frac{b}{2} \rfloor + 1} \geq n$ elements of C are represented by such functions by the pigeon-hole principle. Then delete all other elements of C. ◁

From Properties 5 and 6 we can now deduce that all active valleys of functions representing C are independent. By Property 4, only active valleys can be used for intersection with functions representing D.

Since the active valleys are independent, they form a natural left-to-right sequence. It is convenient to view such a sequence as a word over the alphabet whose symbols are the vertices in the central clique C. We first introduce some related terminology.

We will say that a word w is *m-regular over an alphabet X*, if it contains exactly m occurrences of each symbol from X, and no other symbols. We say that a word w is an *m-immediate repetition over an alphabet X* if it has the form $w = x_1^m x_2^m \cdots x_n^m$, where x^m denotes an m-fold repetition of the symbol $x \in X$, and the n-tuple $(x_1, \ldots, x_n)$ is a permutation of the alphabet X.

We will now introduce the *pattern* of the active valleys, which is a word w over the alphabet C whose i-th symbol w_i is the vertex whose curve contains the i-th active valley from the left. Note that w is m-regular over C, by Property 6.

▶ **Lemma 10.** *For any $m, n \in \mathbb{N}$, there is an $N \in \mathbb{N}$ such that for every m-regular word w over the alphabet $[N]$, there is a set $C \subseteq [N]$ of size n such that the restriction of w to the symbols from C yields a word w' of one of the following two forms:*

1. *there is an integer $j \in [m-1]$ such that w' is a concatenation of two words w'_1, w'_2 where w'_1 is j-regular over C (and hence w'_2 is $(m-j)$-regular over C), or*
2. *the word w' is an m-immediate repetition over the alphabet C.*

Proof. Set $N = mn^2$, and let w be an m-regular word over $[N]$. To each symbol $x \in [N]$ we associate the interval $I_x = [x^i, x^j]$ where x^i, x^j are the indices of the first and last occurrence of x in w.

We then construct an interval graph $G(w)$ as the graph on the vertex set $[N]$ where distinct vertices $x, y \in [N]$ are adjacent if and only if the intervals I_x and I_y intersect. This graph is an interval graph and therefore perfect. In particular, it satisfies $\alpha(G(w)) \cdot \omega(G(w)) \geq N$, where $\alpha(G(w))$ and $\omega(G(w))$ denote the size of the largest independent set and the largest clique of $G(w)$, respectively.

We thus have $\alpha(G(w)) \geq n$ or $\omega(G(w)) \geq mn$. If the former is true, we let C be an independent set of size n in $G(w)$, and we see that Case 2 of the lemma occurs.

On the other hand, if $\omega(G(w)) \geq mn$, we first fix a clique Q of size mn in $G(w)$. The intervals representing the vertices of Q have nonempty intersection by Helly's theorem, and since all these intervals have distinct endpoints by construction, their common intersection is a non-degenerate interval J. Choose a $p \in J$ such that $p+1$ is also in J. Let us call the prefix $w_1 w_2 \cdots w_p$ the *left part* of w, and the suffix $w_{p+1} \cdots w_N$ the *right part* of w. Each symbol $x \in Q$ thus appears both in the left part and in the right part.

For $j \in [m-1]$, let Q_j be the set of those symbols $x \in Q$ that have exactly j occurrences in the left part. Choose the value of j for which the size of Q_j is maximised, and set $C = Q_j$. We then have $|C| \geq |Q|/(m-1) \geq n$, and the restriction of w to the symbols of C satisfies Case 1 of the lemma. ◀

▷ Property 7. We can enforce that the pattern w is a concatenation of $\ell \geq 1$ words $w = I_1 \cdots I_\ell$ where every I_j is an m_j-immediate repetition of the alphabet C, and $\sum_{j=1}^{\ell} m_j = m$. We then call I_j the *j-th interval* of w.

Proof. Apply Lemma 10 iteratively to the intervals in the first case, until the second case is encountered (which is definitely the case if $m = 1$). Note that removing letters from the alphabet of the pattern is equivalent to deleting elements of C. ◁

We may assume that the order of valleys in the first interval I_1 is $(1, \ldots, n)$ by relabeling the vertices in C if necessary.

▷ Property 8. We can enforce that in each of the intervals I_j, the n symbols $1, 2, \ldots, n$ appear either in the increasing order or in the decreasing order.

Proof. For this step choose $N = n^{2^{\ell-1}}$. For each interval consider the sequence of letters of the alphabet, ordered by their occurrences in the interval. In each interval choose one representative of each letter to have a permutation of $(1, \ldots, n)$ as subsequence. We consider just those permutations. By the Erdős-Szekeres Lemma [11], each sequence of n^2 numbers contains a subsequence of n numbers that is either monotonically increasing or monotonically decreasing. With the assumption that the sequence for I_1 is $(1, \ldots, N)$, it is possible to find a subset of $\sqrt{N}$ letters such the permutation for I_2 is increasing or decreasing. By removing all letters not in the subset and repeating the process for each of the intervals $I_3, \ldots, I_\ell$, we end up with at least $N^{\frac{1}{2^{\ell-1}}} = n$ letters such that each interval contains the subsequence $(1, \ldots, n)$ or $(n, \ldots, 1)$ after relabeling. ◁

For a set $I = \{c_1 < c_2 < \cdots < c_t\}$ of t vertices from C, we let $d_I \in D$ denote the vertex in D adjacent to the vertices in I. For such I we further define the *crossing type* of I as the t-tuple $(i(1), i(2), \ldots, i(t)) \in [\ell]^t$ such that for every $j \in [t]$, the curve of d_I has a peak above a valley of the curve of c_j which occurs in the interval $I_{i(j)}$. The crossing type need not be determined uniquely, in which case we assign one fixed crossing type to each $I \in \binom{C}{t}$ arbitrarily.

▷ Property 9. We can enforce that all t-element subsets of C have the same crossing type.

Proof. The crossing types define a coloring of the set $\binom{C}{t}$, and we apply Ramsey's Theorem to this coloring. ◁

▶ **Lemma 11.** *Let Γ be a $\mathfrak{D}$-representation of $H_t(n)$ satisfying the previously enforced properties. Let I be a t-element subset of C that does not contain any two consecutive integers, and contains neither 1 nor n. Let p be a peak of the curve representing d_I. Then there is at most one vertex $c \in I$ whose curve has a valley below p.*

Proof. Since all valleys are independent, every peak lies above a consecutive set of valleys. Say a peak p of d_I lies above valleys of vertices i and $j > i + 1$ from C. If these valleys are in the same interval, Property 8 makes sure that a valley of the function of $i + 1$ lies in between those two valleys, a contradiction since $i + 1 \notin I$. If the valleys of i and j are in different intervals, then the end of one interval and the beginning of the other are contained in I, which is impossible, since I contains neither 1 nor n. ◀

▶ **Theorem 12.** *For every even $k = 2t$ and every n large enough, $H_t(n) \in \mathfrak{D}^{\nearrow}_k \setminus \mathfrak{D}^{\mathsf{H}}_{k-1}$.*

Proof. We already noted that $H_t(n)$ has a $\mathfrak{D}^{\nearrow}_k$-representation for any n. For contradiction, assume that $H_t(n)$ also has a $\mathfrak{D}^{\mathsf{H}}_{k-1}$-representation for n arbitrarily large.

It follows that for arbitrarily large n, the graph $H_t(n)$ has a $\mathfrak{D}^{\mathsf{H}}_{k-1}$-representation with all the properties listed above. We say that a peak p of a function representing $d \in D$ *collects* an element $c \in C$ if it lies above one of the valleys of f_c. Consider the t-subset $I = \{i_1, \ldots, i_t\} \subseteq C, 0 \leq i_1 - 4 \leq i_2 - 8 \leq \ldots \leq i_t - 4t$. Since I does not contain two consecutive integers, nor both n and 1, every peak of f_{d_I} collects a different element of I

by Lemma 11. Let $(r_1, \ldots, r_t) \in C^t$ be the order of the elements of I given by the order of x-coordinates of the peaks of f_{d_I} that collect them. By Property 8 for each $i \in [\ell]$, we can set

$$\sigma_i := \begin{cases} 1, & \text{if } I_i \text{ has the same order as } I_1 \\ -1, & \text{if } I_i \text{ has the reversed order from } I_1 \end{cases}$$
$$I_L := \{r_i - \sigma_{c_{r_i}}(-1)^i \mid i \in [t]\}$$
$$I_R := \{r_i + \sigma_{c_{r_i}}(-1)^i \mid i \in [t]\}$$

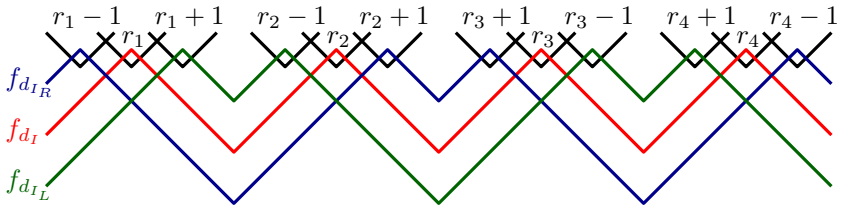


Figure 7 I, I_L and I_R depicted by the corresponding collecting functions. The figure depicts a situation when the intervals containing r_1 and r_2 are increasing, while the intervals containing r_3 and r_4 are decreasing.

We will proceed to show that $\forall\ x \in [0,1] : f_{d_I}(x) > \min_{i\in\{L,R\}} f_{d_{I_i}}(x)$ which is a contradiction since E contains an element that is adjacent to d_I but neither d_{I_L} nor d_{I_R}.

To see this, first note that neither of I_L and I_R contain a pair of consecutive numbers, nor do they contain n and further their elements $r_i \pm 1$ are in the same order as the r_i, since $|r_i - r_j| \geq 4\ \forall i \neq j \in [t]$. Therefore $r_i \pm 1$ is in the same interval as r_i by Property 9. The peak of f_{d_I} collecting r_i is therefore always in between the peaks of $f_{d_{I_L}}$ and $f_{d_{I_R}}$ collecting $r_i \pm \sigma_{c_{r_i}}$. Therefore $f_{d_{I_L}}$ is below f_{d_I} between the peaks collecting r_i and r_{i+1} if i is even, because in this case the valley of $r_i - \sigma_{c_{r_i}}(-1)^i$ is to the left of the valley of r_i and the valley of $r_{i+1} - \sigma_{c_{r_{i+1}}}(-1)^{i+1}$ is to the right of the valley of r_{i+1}. The argumentation is the same for $f_{d_{I_R}}$ if i is odd. Finally the first segment of f_{d_I} is above the function $f_{d_{I_L}}$ because the valley where $r_1 + \sigma_{c_{r_1}}$ is collected is the right neighbor of the valley of r_1. Therefore the function of d_I is lower than either of the functions representing d_{I_R} and d_{I_L} at any point, which completes the contradiction. ◀

For our next theorem, we exploit the properties of the $\mathfrak{D}$-representations of $H_t(n)$ in a different way.

▶ **Lemma 13.** *Let $k = 2t$ be even. For any $\mathfrak{D}^{\nearrow}_k$-representation Γ of $H_t(n)$ with n large enough, there is a line ℓ of slope 1 such that Γ contains a function fully above ℓ as well as a function fully below ℓ.*

Proof. Recall that our proof of Theorem 12 has in fact shown that in a representation of $H_t(n)$, there will be a vertex $v_I \in D$ and two other vertices v_{I_L} and v_{I_R} with the property that each of the t peaks of the curve of v_I is needed to collect a distinct valley of a curve from C, while each of the $t-1$ valleys of v_I between these t peaks is above a valley of v_{I_L} or v_{I_R}. In this situation, we say that v_I is *protected* by v_{I_L} and v_{I_R}. In a $\mathfrak{D}^{\mathsf{H}}_{k-1}$-representation, such a protected vertex yielded a contradiction, since it could not have been crossed by any of its neighbors in E. In a $\mathfrak{D}^{\nearrow}_k$-representation, this merely means that any E-neighbor of a protected vertex must collect its rightmost valley.

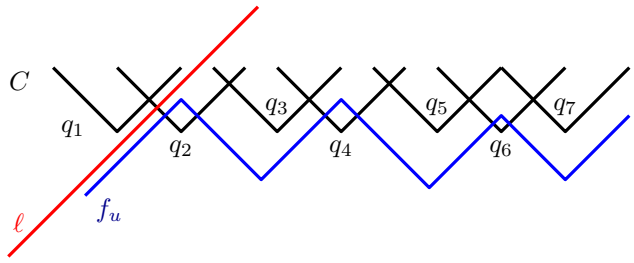


Figure 8 Construction of ℓ in Lemma 13.

Suppose that we have chosen n large enough to be able to construct $2t+1$ protected vertices $w_1, w_2, \ldots, w_{2t+1}$ in D, and let q_i be the rightmost valley in the representation of w_i. Note that the points $q_1, \ldots, q_{2t+1}$ must be independent, since for each of them, there is a neighbor in E that must collect it while avoiding all the others. Suppose that $q_1, \ldots, q_{2t+1}$ were numbered left-to-right (see Figure 8). Consider now a vertex u of E adjacent to $w_2, w_4, \ldots, w_{2t}$. Each peak in the representation of u must collect a different valley among $q_2, \ldots, q_{2t}$, since a single peak collecting more than one of these points would also collect an odd-numbered valley between them.

Let ℓ now be a line of slope $+1$ chosen directly above the leftmost segment of u, so that the valley q_1 is above it. It follows that f_u is below ℓ, while the entire representation of w_1 is above it, since q_1 is the rightmost bend of the representation of w_1. ◀

▶ **Lemma 14.** *Let $k \geq 2$ be even. Then a graph G belongs to $\mathfrak{D}_k^{=}$ if and only if G is the complete join of two graphs G_1 and G_2 from $\mathfrak{D}_k^{\nearrow}$.*

Proof. Let $G \in \mathfrak{D}_k^{=}$. Then the functions where the leftmost segment goes upward induce a graph $G_1 \in \mathfrak{D}_k^{\nearrow}$ and the functions where the leftmost segment goes downward induce a graph $G_2 \in \mathfrak{D}_k^{\nearrow}$ (where the representation is obtained by mirroring at a horizontal line). Note that each function for G_1 crosses each function for G_2 since k is even. Conversely, given $G_1, G_2 \in \mathfrak{D}_k^{\nearrow}$ we obtain the representation of the corresponding graph G by combining the representation of G_1 and the mirrored representation of G_2 (after adding bends such that each function has k bends). ◀

▶ **Theorem 15.** *For k even, there is a graph $G \in \mathfrak{D}_k^{=} \setminus \mathfrak{D}_k^{\nearrow}$.*

Proof. We will show that the conclusion of the theorem holds for the graph G which is the join of two copies of $H_t(n)$, with $t = \frac{k}{2}$. Since we already know that $H_t(n)$ is in $\mathfrak{D}_k^{\nearrow}$ by Theorem 12, Lemma 14 shows that G is in $\mathfrak{D}_k^{=}$. We now prove that G is not in $\mathfrak{D}_k^{\nearrow}$.

Suppose that we are given a $\mathfrak{D}_k^{\nearrow}$-representation Φ of G. Let G_1 and G_2 denote the two copies of $H_t(n)$ whose join is G. By Lemma 13, for each G_i there is a line ℓ_i of slope $+1$ such that in Φ there are curves representing vertices of G_i both fully above and fully below ℓ_i. Suppose without loss of generality that ℓ_1 is above ℓ_2. Then G_1 has a vertex v_1 represented by a curve f_1 above ℓ_1, and G_2 has a vertex v_2 represented by a curve f_2 below ℓ_2. In particular, f_1 and f_2 are disjoint, contradicting the fact that v_1 and v_2 are adjacent in G. ◀

▶ **Theorem 16.** *For $k \geq 2$ even, if G is an arbitrary graph in $\mathfrak{D}_k^{=} \setminus \mathfrak{D}_k^{\nearrow}$, then the following holds.*

1. $G + K_1 \in \mathfrak{D}_k^{\leqslant} \setminus \mathfrak{D}_k^{=}$.
2. $G + G \in \mathfrak{D}_k^{\mathsf{H}} \setminus \mathfrak{D}_k^{\vdash}$.

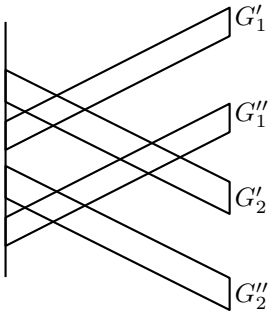


Figure 9 A $\mathfrak{D}_k^{\vdash}$-representation of H from the third part of Theorem 16. Representations of G'_1, G'_2, G''_1, G''_2 can be placed in corresponding infinite parallelograms.

3. *Let us decompose G into G_1 and G_2 as in Lemma 14, then take two copies G'_1, G''_1 of G_1 and two copies G'_2 and G''_2 of G_2, and construct a graph H by adding all edges between G'_1 and G'_2, between G'_2 and G''_1, and between G''_1 and G''_2. Then $H \in \mathfrak{D}_k^{\vdash} \setminus \mathfrak{D}_k^{\leq}$.*

Proof. Fix a graph $G \in \mathfrak{D}_k^{=} \setminus \mathfrak{D}_k^{\nearrow}$. Since $G \notin \mathfrak{D}_k^{\nearrow}$, any $\mathfrak{D}_k^{=}$-representation Γ of G contains at least one function f_Γ whose leftmost segment has slope $+1$ and at least one function g_Γ, whose leftmost segment has slope -1. Since k is even, any function starting with an increasing segment must cross any function starting with a decreasing one, in particular G is connected.

To prove the first part of the theorem, assume that $G + K_1$ can be represented in $\mathfrak{D}_k^{=}$ and fix one such representation Φ. Let $\Gamma' \subset \Phi$ be the set of functions in Φ corresponding to G and let h be the function in Φ representing the single vertex in K_1. Then, without loss of generality, the leftmost segment of h has slope -1 and therefore h crosses $f_{\Gamma'}$, contradicting the assumption that Φ is a representation of $G + K_1$. On the other hand, we can obtain a $\mathfrak{D}_k^{\leq}$-representation of $G + K_1$ by placing a function consisting of two pieces with slopes -1 and $+1$ very far above Γ'.

The second part of the theorem is already covered by Theorem 9.

To prove the last part of the theorem, we assume that H can be represented in $\mathfrak{D}_k^{\leq}$ with a representation Φ. Then, let $f \in \Phi$ be a function representing a vertex in G'_1, we can assume that the leftmost segment of f has slope $+1$. Further, by our choice of G and H, there is at least one other function $g \in \Phi$, representing a vertex of either G''_1 or G''_2 which has leftmost segment with slope -1. Since k is even, f and g cross each other. But this is impossible since vertices in G'_1 are not adjacent to any vertices in G''_1 nor to any vertices in G''_2, so Φ cannot be a representation of H. On the other hand, in Figure 9 we explain how to obtain a $\mathfrak{D}_k^{\vdash}$-representation of H. ◀

4 General setting

This section is concerned with the relationship between the classes $\mathfrak{G}_k^{-}, \mathfrak{G}_k^{\vdash}$ and $\mathfrak{G}_k^{\mathsf{H}}$. The general setting allows for arbitrary slopes, which makes more graphs representable. In fact, you need about half as many bends for similar graphs. However it also turns the investigation quite difficult. Surprisingly, the bounded and grounded representation model do not produce different graph classes though.

▶ **Theorem 17.** *For any $k \in \mathbb{N}_0$, $\mathfrak{G}_k^{\vdash} = \mathfrak{G}_k^{\mathsf{H}}$.*

Proof. Since any number of functions defined on the positive reals can be restricted to a large enough interval without missing any crossings or bends, $\mathfrak{G}_k^{\vdash} \subseteq \mathfrak{G}_k^{\mathsf{H}}$.

For the other inclusion, let $G \in \mathfrak{G}_k^{\mathsf{H}}$. We use a projective transformation that transforms a $\mathfrak{G}_k^{\mathsf{H}}$ representation of G to a $\mathfrak{G}_k^{\vdash}$ representation of G as follows. Suppose the $\mathfrak{G}_k^{\mathsf{H}}$-representation is in the *source plane* embedded in $\mathbb{R}^3$ as the plane $\{(x, y, 0)\}$ with the lines $x = 0$ and $x = 1$ bounding the representation with domain $(0, 1]$. We fix the *perspective point* $P = (0, 0, 1)$ above the source plane, and fix the target plane $\{(-1, y, z)\}$. We now map each point from the source plane to the target plane by projecting it via a line passing through P. Explicitly, this maps a point $(x, y, 0)$ to a point $(-1, -y/x, 1 + 1/x)$. From the construction, one can verify that this preserves collinearity and changes x-monotone curves in the strip $0 < x \leq 1$ into z-monotone ones in the half-plane in the target plane with $2 \leq z < +\infty$. Similarly, each x-monotone line-segment with an open end in line $x = 0$ is transformed into a z-monotone ray with arbitrarily large z-values. On the other hand, observe that each x-monotone line-segment that does not end in line $x = 0$ is transformed to another z-monotone line-segment. Thus, we indeed obtain a $\mathfrak{G}_k^{\vdash}$ representation of the same graph. ◀

As in the diagonal setting, we can observe that the remaining classes form a hierarchy.

▶ **Theorem 18.** *For any $k \in \mathbb{N}_0$, we have $\mathfrak{G}_k^{-} \subseteq \mathfrak{G}_k^{\mathsf{H}} \subseteq \mathfrak{G}_{k+1}^{-}$.*

Proof. The first inclusion follows from Proposition 1. For the other inclusion, let $G \in \mathfrak{G}_k^{\mathsf{H}}$. By Theorem 17 we can fix some representation of G in $\mathfrak{G}_k^{\vdash}$. We now extend all the functions of G to $\mathbb{R}$ by another constant segment on the negative reals in a continuous way. This is a representation of $G \in \mathfrak{G}_{k+1}^{-}$, since we added at most one bend and no functions intersect on the negative reals. ◀

For very small values of k, we can see that there are strict inclusions between $\mathfrak{G}_k^{-}$ and $\mathfrak{G}_k^{\mathsf{H}}$. In particular, it is easy to see that the class $\mathfrak{G}_0^{-}$ is just the class of multipartite graphs and is therefore strictly included in $\mathfrak{G}_0^{\mathsf{H}}$, which is exactly the class of permutation graphs [22]. We cannot exactly characterize the graphs in $\mathfrak{G}_1^{-}$, but we can show that for every $n \in \mathbb{N}$, the cocomparability graph of the *standard example* S_n [24] admits a $\mathfrak{G}_1^{-}$-representation (see Figure 3), and we can therefore conclude that $\mathfrak{G}_0^{\mathsf{H}} \subsetneq \mathfrak{G}_1^{-}$. We remark that we we believe that it is possible to prove the strict inclusions $\mathfrak{G}_1^{-} \subsetneq \mathfrak{G}_1^{\mathsf{H}} \subsetneq \mathfrak{G}_2^{-} \subsetneq \mathfrak{G}_2^{\mathsf{H}}$, but the argument is tedious and the graphs witnessing the strictness of the inclusions do not generalize to higher values of k. Therefore we do not include these results. Our next aim is to prove that $\mathfrak{G}_k^{\mathsf{H}}$ is a strict subclass of $\mathfrak{G}_{k+1}^{-}$ for all $k \geq 2$.

The graph $G_t \equiv G_t(n)$ consists of three cliques $A, B \cup C, D$, where B and C have size n while A and D are t-extension of B and C, respectively. There are no edges in G_t other than those implied by the previous rules.

▶ **Theorem 19.** *For any $t \geq 4$ and n large enough, the graph $G_t(n)$ is in $\mathfrak{G}_{t-1}^{-} \setminus \mathfrak{G}_{t-2}^{\mathsf{H}}$.*

Proof. The proof that $G_t(n) \in \mathfrak{G}_{t-1}^{-}$ can be found in the full version of our paper, see Figure 10 for illustration.

Toward a contradiction, assume that $G_t \in \mathfrak{G}_{t-2}^{\mathsf{H}}$ and fix a representation. Let $a \in A, d \in D$. Since the corresponding functions f_a and f_d do not intersect, without loss of generality, we may assume that $f_a \geq f_d$. Since A and D are nonadjacent, $f_a \geq f_d$ holds not only for these specific functions, but for all $a \in A, d \in D$.

We will delete elements of B and all elements of A adjacent to it and an equivalent amount of elements from C and D in the same manner similarly to Theorem 12. The goal is for every $n \in \mathbb{N}$ to find $N \in \mathbb{N}$ in order to arrive at a representation of a $G_t(n)$ subgraph of

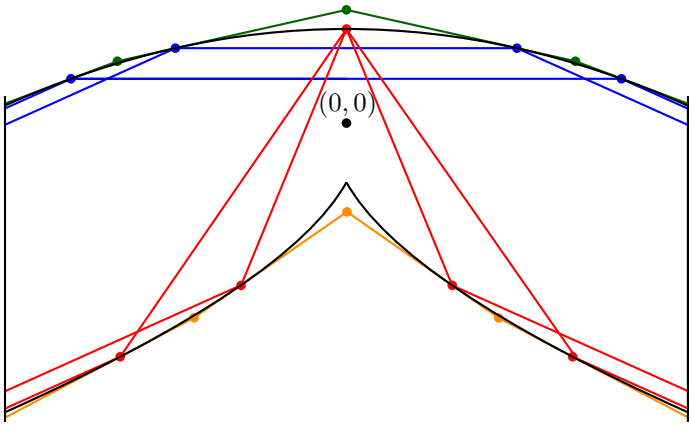


Figure 10 Representation of G_4 with only two functions in B (blue) and C (red) that are touched twice in both (non-middle) bends by one function each in A (green) and D (orange). In a proper representation, different functions of B or C would be touched by the functions of A and D in the positive and negative reals.

$G_t(N)$ that fulfills some property, then prove that such a representation cannot exist. In order to state the property, we need to define the so-called first level of an arrangement of functions, which is the graph of the second highest function at every point, formally $L_1(f_1, \ldots, f_m)(x) = \max\{y \in \mathbb{R} \colon |\{i \in [m] \colon f_i(x) \geq y\}| = 2\}$

▷ Property 10. No function in A crosses the first level of the functions in B.

Proof. Agarwal, Aronov, Chan, and Sharir [2] prove that the first level of a set of s segments has complexity $\mathcal{O}(s\,\alpha(s))$, where α is the inverse Ackermann function and complexity means the number of intersection points plus discontinuities. Since B contains only continuous functions, there are no discontinuities. The number of segments of some subset of k functions of B is $(t-1)k$. Therefore, the number of intersection points and bends along the first level is in $\mathcal{O}(tk\,\alpha(tk))$ and so is the number of segments between them.

We define a graph H with vertex set B where we add an edge between b_i and b_j if there exists $x \in [0,1]$, such that $f_{b_i}(x)$ and $f_{b_j}(x)$ are the two largest values among $\{f_b(x) \mid b \in B'\}$. In this case one of these functions contributes a segment to the first level and all x-values of this segment qualify as witnesses to the existence of the edge b_ib_j. Thus, the number of edges of H is $\mathcal{O}(tn\,\alpha(tn))$ so its average degree is in $\mathcal{O}(t\,\alpha(tn))$. By Turán's Theorem H contains an independent set of size $\Omega\left(\frac{n}{t\,\alpha(nt)}\right)$. Restricting the set B to this independent set, the property follows, since if any function in A crosses the first level, it drops below two or more functions of B, which consequently have an edge in H, a contradiction. ◁

We are particularly interested in the upper envelope or maximum of all functions f_b representing $b \in B$. A *chunk* of the upper envelope is a connected subset of it that is contained in the graph of a single function, i.e. a connected subset of $\{(x, f_b(x)) \in \mathbb{R}^2 \mid f_b(x) = \max_{b' \in B} f_{b'}(x)\}$. A *chunk set* is a set of t chunks. A chunk set is *crossable*, if there is a function f_a representing some vertex $a \in A$ that intersects all of its chunks. A set of functions $\{f_{b_1}, \ldots, f_{b_t}\}$ is *crossable* if there exists a crossable chunk set $\{c_1, \ldots, c_t\}$, such that for all $i \in [t]$, c_i is part of f_{b_i}. By the definition of A and Property 1, every set of t functions is crossable, so for contradiction, we are going to show that this is not true. We will even be able to show that the number of crossable chunk sets is less than $\binom{n}{t}$.

To do so, we group the chunk sets using a parameter $\ell = \lfloor \sqrt{c} \rfloor$ where c is the number of chunks. It is a well-known fact that the upper envelope of s segments has complexity at most $\mathcal{O}(s\,\alpha(s))$; see for instance Sharir and Agarwal [23], chapter 2. So the total number of chunks is $c \in \mathcal{O}(nt\,\alpha(nt))$. We can therefore identify the chunks with numbers in $[c]$ and the chunk sets with sets in $\binom{[c]}{t}$. An *early* chunk set includes a number $< \ell$. A *short* chunk set includes two numbers of difference $< \ell$. A *group* is a set of disjoint chunk sets of the form $\{\{i_1 - j, \ldots, i_{t-1} - j, c - k\ell - j\} \mid 0 \le j < l\}$ for some $k \in \mathbb{N}_0$. Note that the *maximum set* $\{i_1, \ldots, i_{t-1}, c - k\ell\}$ has maximum $c - k\ell$ and is neither short nor early. Further note that each chunk set is short, early or in a group. Two distinct chunk sets contained in the same group are not only disjoint but alternating, that is, if the smallest number of them is in one of the sets, then the second smallest is in the other and so on. We will use this fact and the following observation in the subsequent claim:

▶ **Observation 20.** *Two piecewise linear functions with $t-2$ bends cross $\le 2t-3$ times.*

Proof. They define at most $2t-3$ intervals between the $2t-4$ bends (and on either end), where both functions are (affine) linear, thus only cross once. ◁

▷ Claim 21. At most $4t^2$ of the chunk sets in a group are crossable.

Proof. Consider the subset $S = \{s_1, \ldots, s_k\}$ of crossable chunk sets of a group and fix $i, j \in [k]$ as well as functions $f_1, \ldots, f_k$ that witness the crossability of $s_1, \ldots, s_k$. If the sets of chunks intersected by f_i and f_j are disjoint, observe that between their intersections with any two chunks $c_i \in s_i, c_j \in s_j$, f_i and f_j have to cross by the intermediate value theorem. Since the chunks alternate, there are at least $2t-1$ such crossings, a contradiction to the above observation. Therefore a chunk in s_i is crossed by f_j or vice-versa. We construct a tournament on S, such that (s_i, s_j) is an arc in the former case and (s_j, s_i) is an arc otherwise. However we can also deduce from the observation that any function in A crosses at most $t \cdot (2t-3) \le 2t^2 - 1$ chunks, because a function in A only crosses t functions of B, each of which is crossed at most $2t-3$ times. For the tournament on S, this means that the indegree of any vertex is at most $2t^2 - 1$, so the number of edges is $\binom{k}{2} \le (2t^2-1)k \Rightarrow k \le 4t^2$. ◁

Using the above claim, we can now prove that the number of crossable chunk sets is too small. Recall that each chunk set is early, short or in a group. For the early chunk sets, we have $\le \ell$ possibilities to choose the smallest number and $\le \binom{c}{t-1}$ possibilities to choose the rest. For the short chunk sets, we have $\le \ell c$ possibilities to place two numbers of distance $< \ell$ and $\le \binom{c}{t-2}$ possibilities to choose all other numbers. Observe that there are $\le \binom{c}{t}/\ell$ groups by the restriction of the maximum to some $c - \ell k$. It follows that the number of early chunk sets N_E is at most $\ell\binom{c}{t-1} \le \ell c^{t-1}$ and the number of short chunk sets N_S is at most $\ell c\binom{c}{t-2} \le \ell c^{t-1}$. For the total number of crossable chunk sets N_c we get the upper bound

$$N_c \le N_E + N_S + \frac{4t^2}{\ell}\binom{c}{t} \le 2\ell c^{t-1} + \frac{4t^2 c^t}{\ell} \in \Theta(t^2 c^{t-1/2}) \subseteq \mathcal{O}(t^2 (nt\,\alpha(nt))^{t-1/2}) \subseteq o_t(n^t).$$ ◀

5 Concluding remarks and open problems

Our investigation of the bend number and the related graph classes leads to several open questions. We highlight the ones that we consider the most interesting.

We first present some combinatorial problems. We introduced the graph classes $\mathfrak{G}_k^-$ and $\mathfrak{G}_k^{\mathsf{H}}$, and while we can prove that $\mathfrak{G}_k^- \subseteq \mathfrak{G}_k^{\mathsf{H}}$, we do not know weather these inclusions are strict:

▶ **Conjecture 22.** *For any $k \in \mathbb{N}_0$, $\mathfrak{G}_k^-$ is a proper subset of $\mathfrak{G}_k^{\mathsf{H}}$.*

For $k = 0$, the conjecture holds, since $\mathfrak{G}_0^-$ is the class of complete multipartite graphs, while $\mathfrak{G}_0^{\mathsf{H}}$ is the class of permutation graphs. For $k = 1$ and $k = 2$ we can also verify that the conjecture holds: in fact, using the graphs $G_t(n)$ used in the proof of Theorem 19, we are able to verify that $G_2(n) + G_2(n)$ is in $\mathfrak{G}_1^{\mathsf{H}} \setminus \mathfrak{G}_1^-$, while $G_3(n) + G_3(n)$ is in $\mathfrak{G}_2^{\mathsf{H}} \setminus \mathfrak{G}_2^-$. Theorem 19 shows this does not straightforwardly generalize to higher k. Verifying Conjecture 22 would complete our understanding of the hierarchy induced by the classes $\mathfrak{G}_k^-$ and $\mathfrak{G}_k^{\mathsf{H}}$.

Another natural question is to determine the maximum number of bends needed to represent every poset of size n. In the diagonal setting, by rotating the representation by 45 degrees, we can view it as a VPG-representation with at most $(k+2)n$ bends and endpoints. Since rows and columns without these points are unnecessary the whole arrangement of curves can be embedded to an integer grid of size $\mathcal{O}(kn) \times \mathcal{O}(kn)$, thus encoded using $\mathcal{O}(kn \log(kn))$ bits. Hence there are no more than $2^{\mathcal{O}(kn \log(kn))}$ cocomparability graphs in $\mathfrak{D}_k^{\mathsf{H}}$. Since there are $2^{\Omega(n^2)}$ n-vertex cocomparability graphs, some of them must require $\Omega\left(\frac{n}{\log n}\right)$ bends per curve in a diagonal representation. We do not know if this bound is tight, and we are not aware of a similar lower bound for $\mathfrak{G}_k^{\mathsf{H}}$-representations. However enumeration of intersection graphs has been investigated for various classes of intersection graphs [1, 10, 12, 15, 18, 19, 26, 27], and in particular for some natural subclasses of cocomparability graphs, such as permutation graphs and interval graphs [1, 12, 18, 26, 27]. Since our classes naturally refine the hierarchy of subclasses of cocomparability graphs, determining their size could be an interesting problem.

Natural questions also arise on the complexity-theoretic side, mainly determining if any of our classes can be recognized in polynomial time may be interesting. It can be shown that the recognition problem for "diagonal" classes $\mathfrak{D}_k^{\nearrow}, \mathfrak{D}_k^{=}, \mathfrak{D}_k^{\leq}, \mathfrak{D}_k^{\vdash}, \mathfrak{D}_k^{\mathsf{H}}$ is in NP. However, for the classes $\mathfrak{G}_k^-$ and $\mathfrak{G}_k^{\mathsf{H}}$, even membership in NP is not clear. Note that both $\mathfrak{D}_1^{\nearrow}$ and $\mathfrak{G}_0^{\mathsf{H}}$ are equal to the class of permutation graphs, which are known to be recognisable in polynomial time. We expect that recognition will turn out to be NP-hard for classes with higher bend-number, but it may be interesting to pin down the boundary of tractable recognition.

Lastly, there are several well-studied computational problems that remain NP-hard when restricted to cocomparability graphs but can be solved efficiently for interval or permutation graphs [5, 13, 20, 25]. It would be interesting to determine where exactly these problems become NP-hard, and our classes give a natural setting to investigate this.

References

1 Hüseyin Acan, Sankardeep Chakraborty, Seungbum Jo, and Srinivasa Rao Satti. Succinct data structures for families of interval graphs. In *Algorithms and data structures*, volume 11646 of *Lecture Notes in Comput. Sci.*, pages 1–13. Springer, Cham, 2019. doi:10.1007/978-3-030-24766-9_1.

2 Pankaj K. Agarwal, Boris Aronov, Timothy M. Chan, and Micha Sharir. On levels in arrangements of lines, segments, planes, and triangles. *Discrete &; Computational Geometry*, 19:315–331, 1998. doi:10.1007/PL00009348.

3 Andrei Asinowski, Elad Cohen, Martin Charles Golumbic, Vincent Limouzy, Marina Lipshteyn, and Michal Stern. String graphs of k-bend paths on a grid. In *LAGOS'11—VI Latin-American Algorithms, Graphs and Optimization Symposium*, volume 37 of *Electron. Notes Discrete Math.*, pages 141–146. Elsevier Sci. B. V., Amsterdam, 2011. doi:10.1016/j.endm.2011.05.025.

4 Andrei Asinowski, Elad Cohen, Martin Charles Golumbic, Vincent Limouzy, Marina Lipshteyn, and Michal Stern. Vertex intersection graphs of paths on a grid. *J. Graph Algorithms Appl.*, 16(2):129–150, 2012. doi:10.7155/jgaa.00253.

5 Andreas Brandstädt and Dieter Kratsch. *On the restriction of some NP-complete graph problems to permutation graphs*, pages 53–62. Springer-Verlag, 1985. doi:10.1007/bfb0028791.

6 Dibyayan Chakraborty, Sandip Das, Joydeep Mukherjee, and Uma Kant Sahoo. Bounds on the bend number of split and cocomparability graphs. *Theory Comput. Syst.*, 63(6):1336–1357, 2019. doi:10.1007/s00224-019-09912-4.

7 Dibyayan Chakraborty, Kshitij Gajjar, and Irena Rusu. Recognizing geometric intersection graphs stabbed by a line. *Theoret. Comput. Sci.*, 995:Paper No. 114488, 16, 2024. doi:10.1016/j.tcs.2024.114488.

8 Steven Chaplick, Vít Jelínek, Jan Kratochvíl, and Tomáš Vyskočil. Bend-bounded path intersection graphs: Sausages, noodles, and waffles on a grill. In Martin Charles Golumbic, Michal Stern, Avivit Levy, and Gila Morgenstern, editors, *Graph-Theoretic Concepts in Computer Science*, pages 274–285, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. doi:10.1007/978-3-642-34611-8_28.

9 Elad Cohen, Martin Charles Golumbic, William T. Trotter, and Ruidong Wang. Posets and VPG graphs. *Order*, 33(1):39–49, 2016. doi:10.1007/s11083-015-9349-9.

10 Lars Eirik Danielsen and Matthew G. Parker. Interlace polynomials: Enumeration, unimodality and connections to codes. *Discrete Applied Mathematics*, 158(6):636–648, March 2010. doi:10.1016/j.dam.2009.11.011.

11 Paul Erdős and George Szekeres. A combinatorial problem in geometry. *Compositio Mathematica*, 2:463–470, 1935. URL: http://eudml.org/doc/88611.

12 Aysel Erey, Zachary Gershkoff, Amanda Lohss, and Ranjan Rohatgi. Characterization and enumeration of 3-regular permutation graphs. *Rocky Mountain J. Math.*, 55(1):89–100, 2025. doi:10.1216/rmj.2025.55.89.

13 Hiroshi Eto, Takehiro Ito, Eiji Miyano, Akira Suzuki, and Yuma Tamura. Happy set problem on subclasses of co-comparability graphs. *Algorithmica*, 85(11):3327–3347, December 2022. doi:10.1007/s00453-022-01081-0.

14 Stefan Felsner, Kolja Knauer, George B. Mertzios, and Torsten Ueckerdt. Intersection graphs of L-shapes and segments in the plane. *Discrete Appl. Math.*, 206:48–55, 2016. doi:10.1016/j.dam.2016.01.028.

15 Jacob Fox, János Pach, and Andrew Suk. Enumeration of intersection graphs of x-monotone curves. In *32nd International Symposium on Graph Drawing and Network Visualization (GD 2024)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.GD.2024.4.

16 Martin Charles Golumbic, Doron Rotem, and Jorge Urrutia. Comparability graphs and intersection graphs. *Discrete Math.*, 43(1):37–46, 1983. doi:10.1016/0012-365X(83)90019-5.

17 Daniel Gonçalves, Lucas Isenmann, and Claire Pennarun. Planar graphs as $\llcorner$-intersection or $\llcorner$-contact graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 172–184. SIAM, Philadelphia, PA, 2018. doi:10.1137/1.9781611975031.12.

18 Jun Kawahara, Toshiki Saitoh, Hirokazu Takeda, Ryo Yoshinaka, and Yui Yoshioka. Efficient non-isomorphic graph enumeration algorithms for subclasses of perfect graphs. In *WALCOM: algorithms and computation*, volume 13973 of *Lecture Notes in Comput. Sci.*, pages 151–163. Springer, Cham, [2023] ©2023. doi:10.1007/978-3-031-27051-2_14.

19 Jan Kynčl. Improved enumeration of simple topological graphs. *Discrete & Computational Geometry*, 50(3):727–770, September 2013. doi:10.1007/s00454-013-9535-8.

20 Chang Maw-Shang. Weighted domination of cocomparability graphs. *Discrete Applied Mathematics*, 80(2–3):135–148, December 1997. doi:10.1016/s0166-218x(97)80001-7.

21 Matthias Middendorf and Frank Pfeiffer. The max clique problem in classes of string-graphs. *Discrete Math.*, 108(1-3):365–372, 1992. doi:10.1016/0012-365X(92)90688-C.

22 Nikita Nesterenko. Structural and complexity aspects of intersection representations. Bsc. thesis, Charles University, Prague, 2024. Available at http://hdl.handle.net/20.500.11956/193177.

23 Micha Sharir and Pankaj K. Agarwal. *Davenport-Schinzel Sequences and their Geometric Applications*. Cambridge University Press, 1995.

24 William T. Trotter. *Partially ordered sets*, pages 433–480. MIT Press, Cambridge, MA, USA, 1996.

25 Kuo-Hui Tsai and Wen-Lian Hsu. Fast algorithms for the dominating set problem on permutation graphs. *Algorithmica*, 9(6):601–614, June 1993. doi:10.1007/bf01190158.

26 Kazuaki Yamazaki, Toshiki Saitoh, Masashi Kiyomi, and Ryuhei Uehara. Enumeration of nonisomorphic interval graphs and nonisomorphic permutation graphs. *Theoretical Computer Science*, 806:310–322, February 2020. doi:10.1016/j.tcs.2019.04.017.

27 Joyce C. Yang and Nicholas Pippenger. On the enumeration of interval graphs. *Proc. Amer. Math. Soc. Ser. B*, 4:1–3, 2017. doi:10.1090/bproc/27.

Towards a Better Understanding of Graph Perception in Immersive Environments

Lin Zhang[1]
University of Stuttgart, Germany

Yao Wang[2]
University of Stuttgart, Germany

Ying Zhang
University of Konstanz, Germany

Wilhelm Kerle-Malcharek
University of Konstanz, Germany

Karsten Klein
University of Konstanz, Germany

Falk Schreiber
University of Konstanz, Germany
Monash University, Melbourne, Australia

Andreas Bulling
University of Stuttgart, Germany

Abstract

As Immersive Analytics (IA) increasingly uses Virtual Reality (VR) for stereoscopic 3D (S3D) graph visualisation, it is crucial to understand how users perceive network structures in these immersive environments. However, little is known about how humans read S3D graphs during task solving, and how gaze behaviour indicates task performance. To address this gap, we report a user study with 18 participants asked to perform three analytical tasks on S3D graph visualisations in a VR environment. Our findings reveal systematic relationships between network structural properties and gaze behaviour. Based on these insights, we contribute a comprehensive eye tracking methodology for analysing human perception in immersive environments and establish eye tracking as a valuable tool for objectively evaluating cognitive load in S3D graph visualisation.

2012 ACM Subject Classification Human-centered computing → Empirical studies in visualization; Human-centered computing → Virtual reality; Human-centered computing → Visual analytics

Keywords and phrases Stereoscopic 3D, Graph Visualisation, Eye Tracking, Graph Perception

Digital Object Identifier 10.4230/LIPIcs.GD.2025.11

Supplementary Material *Dataset*: `https://doi.org/10.18419/DARUS-5259`
archived at `swh:1:rel:fbacfd901cd31509b02fda86ce92972dace66205`

Funding This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 251654672 – TRR 161 (Project A07, A09, and D04).

Acknowledgements This project started at the SFB-TRR 161 Hackathon 2025.

[1] equal contribution
[2] equal contribution, corresponding author



© Lin Zhang, Yao Wang, Ying Zhang, Wilhelm Kerle-Malcharek, Karsten Klein, Falk Schreiber, and Andreas Bulling;
licensed under Creative Commons License CC-BY 4.0



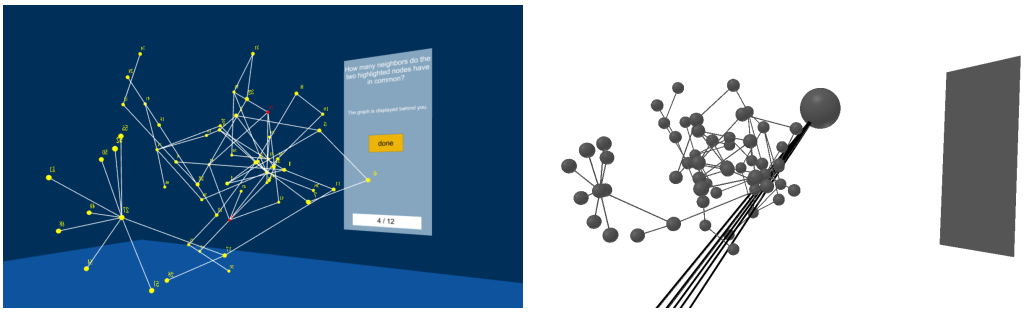


Figure 1 Left: Our immersive environment displaying a 3D graph with highlighted nodes shown in red for the common neighbours task (T_{CN}). The question panel displays the task instruction and progress indicator. **Right**: Visualisation of the perspective reconstruction and ray casting. The large sphere represents the camera position. A cone of nine rays emanates from the camera origin. The black rectangular plane represents the question panel.

1 Introduction

Graph visualisation in stereoscopic 3D (S3D) is gaining significant attention, with an increasing number of research publications focusing on visualisation and interaction methods that enhance graph analysis through immersive environments. Early studies in the 1990s indicated the beneficial effects of head-tracking S3D displays on graph analysis performance [49]. Subsequent research has reaffirmed these findings across various tasks and technologies, including community detection [13] and augmented reality (AR) applications [2]. The advent of affordable, high-quality head-mounted display (HMD) systems and the emergence of Immersive Analytics as a research domain have contributed to a substantial increase in studies addressing not only graphs but also a diverse array of abstract data types [23, 19, 12].

Despite this progress, the extensive design space encompassing technology, graph representation, environmental characteristics, movement, interaction, and navigation has resulted in only a partial understanding of the effects of S3D graph visualisation. Current evidence from studies covers only a limited portion of this design space. The insights gathered thus far suggest that there is no general superiority of S3D over 2D visualisation [13, 53]; rather, performance appears to be highly dependent on various factors, including graph characteristics, representation methods, task types, and interaction support [19, 11, 14, 25, 42]. For some settings also differences between accuracy and response time or even trade-offs have been reported [13, 25]. Consequently, it is crucial to investigate how graph reading and task-solving strategies differ between S3D and 2D and why these differences exist. While these questions are complex and cannot be fully addressed in a single project, this study aims to advance our understanding by examining task-related graph perception in S3D. To explore potential issues, misperceptions, or advantages associated with S3D, we employ eye-tracking technology to analyse graph reading behaviour.

Although recent studies have begun to investigate perception in 2D graph visualisations (e.g. [32, 41, 26, 48]), research on S3D graph perception remains limited compared to traditional 2D setups. Experimental evidence regarding analysis performance presents a mixed picture for 2D versus S3D, with some tasks and scenarios favouring 2D and others demonstrating advantages for S3D [13, 25, 42, 11]. Investigating the perception and gaze behaviour of analysts in S3D can help identify potential challenges and obstacles associated with 3D visualisations, thereby contributing to the development of improved quality measures and layout methods specifically tailored for S3D.

A key component of this investigation is the collection and analysis of eye-tracking data. In addition to achieving the aforementioned goals, eye tracking can provide insights into reading behaviour and help identify or validate task-solving strategies and comprehension models, which are often derived from self-reports [3, 27]. An important consideration for S3D is the applicability of existing quality criteria and the potential need for new criteria. While numerous quality metrics exist for 2D graph layouts, they do not necessarily translate directly to 3D due to factors such as perspective dependency, particularly concerning edge crossings. Although eye tracking has been underutilised in graph visualisations, it has proven effective in investigating readability factors to enhance quality measures, such as edge crossing angles [15], and in comparing layout effects [40, 39]. However, eye tracking for S3D visualisations presents unique challenges, including depth perception issues, occlusion, and perspective changes. Consequently, there is significantly less research employing eye tracking in S3D graph visualisation compared to 2D.

In this study, we make a significant contribution by exploring the use of eye tracking in one of the first investigations of its kind, aiming to gain insights into gaze behaviour during task-solving. This represents an initial step toward identifying and differentiating behaviour categories among individuals, as well as confirming or refuting hypotheses regarding task-solving strategies, readability obstacles, and quality metrics.

2 Related Work

Our work investigates the use of eye tracking to analyse human perception and task solving for graph visualisation in S3D. We give an overview of previous works where we focus on the specific aspects of perception and graph visualisation, in particular for VR and AR. Stereoscopic 3D graph visualisations have been investigated over several decades now, with rather sparse coverage before the advent of affordable mainstream VR hardware and a strong increase shortly after. Due to the rich literature on general aspects of 3D graph visualisation and for applications, we refer to corresponding surveys [19, 23, 12]. For a general overview on eye tracking in 3D environments, see the survey on gaze interaction and eye tracking in XR HMDs [38] and the review on eye tracking visualisation in 3D [45].

Human Graph Perception

Previous studies examined how graph layout impacts readability and perception. Huang et al. [15, 16] found that eye movement smoothness and readability depend on edge-crossing angles. Sharif and Maletic [40] showed clustered UML layouts improve class search performance. Force-directed layouts were found most readable in node/path tasks [39], while tapered edges aided trajectory perception [35]. Kypridemou et al. [26] and Soni et al. [41] examined the perception of graph characteristics such as density in dependence of the graph layout used, with mixed results on the significance of the layout impact. Mooney et al. [33] designed a perceptual study to train novices to tell "stress" in graph drawing. Zhao et al. [56] investigated degree centrality perception and identified several impact factors. The human-centred experiments on node–link diagrams and visual complexity were surveyed in [54]. With eye tracking technology, Chen et al. explored reading strategies for graphs laid out on a flattened Torus topology and successfully distinguished different strategies based on gaze data [6]. Moreover, Steichen et al. [43] and Wang et al. [47] used gaze data to predict visualisation tasks and user cognitive abilities. Okoe et al. implemented gaze interaction techniques to facilitate graph reading tasks [36].

In the field of immersive analytics, Whitlock et al. [52] emphasised the importance of tailoring visualisations to human cognitive processes in VR. Yang et al. [53] showed that visual cues such as motion, shading, and perspective significantly influence accuracy in HMD-based scatterplot tasks. Sun et al. [44] provided a comprehensive review of visual perception models for immersive displays, advocating for designs aligned with perceptual characteristics. Tory et al. [46] used eye tracking to compare 2D and 3D view arrangements, finding higher engagement with 3D, especially when centrally positioned.

Overall, prior work shows that graph layout and eye tracking are both key to understanding visual perception. In immersive environments, additional cues such as depth, motion, and spatial arrangement further shape how graphs are read. Eye tracking in VR thus offers a unique opportunity to capture human attention shifts, helping us to understand the exploration strategy during graph reading.

Graph Visualisation in Virtual and Augmented Reality

Marques et al. [29] demonstrated that augmented reality offers superior spatial perception and immersion for unit visualisation. Graph visualisation in immersive environments has been an active research field, and comprehensively surveyed [19, 23, 12]. Most publications have targeted specific aspects of the design for graph visualisation, such as the encoding [5], the perspective and viewpoint selection [42], spatial arrangement [11], or the interaction, in particular for navigation [10]. Further investigations were related to specific application use-cases, such as brain activity analysis [24, 8, 17]. Prior works have explored how immersive visualisation techniques impact user performance. Joos et al. [18] tried to quantify the perceptual quality of 3D viewing perspectives by a set of standard quality measures and used the results for user-guidance by mapping on a sphere for viewpoint selection. Kwon et al. [25] found that egocentric rendering and interaction in VR improved response times. Sorger et al. [42] further demonstrated benefits from egocentric perspectives with adaptive local scene layouts. Greffard et al. [13] found that S3D projections aided community detection in complex graphs, though 2D yielded faster responses. S3D also enables richer spatial organisation of multiple structures. Feyer et al. [11] compared 2D, 2.5D, and 3D multi-layer graph layouts in VR, highlighting perceptual trade-offs. Visual encoding remains underexplored in immersive graph environments; Büschel et al. found straight edges preferable, as curved edges impaired performance [4]. Ware and Franck, and Ware and Mitchell [49, 50, 51] have concluded that the combination of stereo and kinetic cues can improve graph reading abilities up to an order of magnitude compared to 2D settings. Belcher et al. [2] compared 2D viewing with AR visualisations using a tangible interface. They observed smaller error rates but higher answering times in AR and confirmed Ware and Franck's conclusions that adding stereo cues alone provides less advantage than kinetic depth cues.

While prior work has explored interaction methods, perceptual cues, and spatial arrangements for immersive graph visualisation, it focused on isolated design aspects or coarse performance metrics such as task accuracy and completion time. Only a few have examined how perceptual mechanisms unfold over time or how users dynamically engage with 3D graph structures during task-solving. Furthermore, limited attention has been given to the methodological challenges of analysing depth-rich, mesh-based environments in stereoscopic VR. Our work complements this literature by introducing a detailed ray-casting-based eye tracking approach that captures fine-grained gaze behaviour, offering a deeper understanding of how task demands and graph topology shape attention in immersive graph visualisations.

3 Study Design

3.1 Graph Selection

For this initial study, our main aim is not to investigate the effects of scale by covering a large range of graph characteristics. Still, we wanted to avoid observing peculiarities based on the structure of only a single graph instance. Thus, three undirected simple graph instances were included in our study: two real-world instances and one generated graph. The size range between 50 and 75 nodes was chosen such that the graphs are neither trivial nor extremely challenging due to visual and cognitive overload [55], e.g. for overview and navigation tasks. The aim is to ensure graph visibility and interpretability and mitigate potential occlusion between nodes at varying depths. In addition, it allows us to investigate the applicability of our methodology without confounding results by effects from extreme cases, e.g. large or extremely dense graphs, which deserve an investigation of their own.

The real-world instances were selected from the Konect collection: the Iceland[3] (Graph_{ic}), a graph of social contacts in Iceland with 75 nodes and 114 edges, and Edit-bmwikiquote[4] (Graph_{ed}), a bipartite edit graph from the Bambara Wikiquote containing users and pages connected by edit events. As the original edit-bmwikiquote graph was unconnected with 84 nodes, we extracted the largest connected component, resulting in 56 nodes and 65 edges for our analysis. The three-dimensional coordinates of nodes in these selected graphs were generated using force-directed simulation algorithms implemented in D3-force[5]. Additionally, we synthesised a third graph (Graph_{sy}) with 50 nodes and 100 edges utilising the Watts-Strogatz model generator in the OGDF library [7]. The graphs differ significantly in several core characterisation measures (mean degree, diameter, mean clustering coefficient, mean shortest path): Graph_{ic} (3.04, 6, 0.286, 5.2), Graph_{ed} (2.32, 11, 0(bipartite), 3.19), Graph_{sy} (4.0, 6, 0.07, 2.86). We subsequently normalised the spatial coordinates of all nodes to position each graph within a $1\,m^3$ cubic volume in the virtual reality environment. The edge lengths remain relatively consistent across the three graphs. The edge lengths for the three graphs are as follows: Graph_{sy} ($\mu = 0.21$, $\sigma = 0.05$), Graph_{ed} ($\mu = 0.22$, $\sigma = 0.08$), and Graph_{ic} ($\mu = 0.20$, $\sigma = 0.06$). Student's T-test comparisons revealed no significant differences in edge lengths across graphs ($p > 0.05$ for all pairwise comparisons).

3.2 Task Selection

Our study tasks contain three commonly performed graph perceptual tasks [31]:

- Task shortest path (T_{SP}): *How long is the shortest path between the two highlighted nodes (in number of edges)?* Two randomly selected nodes are highlighted with a red sphere. The participants must search for the shortest path. All participants see the same highlighted nodes for each graph. The length was controlled between 3 and 5 for a good task complexity. The answer panel shows buttons for all possible answers (0 to 9).
- Task highest node degree (T_{HD}): *Which node has the highest number of edges to other nodes?* We ensure there is a unique node in every graph with the highest degree. For our study, the highest node degree ranged from 7 to 24. The participants must search for the node with the highest degree, and then enter the chosen node label on the answer panel.

[3] http://konect.cc/networks/iceland/
[4] http://konect.cc/networks/edit-bmwikiquote/
[5] https://d3js.org/d3-force

- Task common neighbour (T_{CN}): *How many common neighbours do the two highlighted nodes share?* All participants see the same highlighted nodes for each graph. For our study, the number of common neighbours ranged from 3 to 5. The participants should count the number of neighbours and select the answer from the panel (0 to 9).

3.3 Experiment Design and Setup

3.3.1 VR Environment

The VR environment was implemented with the WebXR Device API[6] and Babylon.js[7] serving as the rendering framework. Participants accessed the study environment through the immersive mode of HMD's native web browser. The experimental procedures were conducted using a Meta Quest 3 HMD integrated with the NeonXR eye tracking system [1] from Pupil Labs. The refreshing rate was higher than 50 Hz during the entire study for a smooth experience. The study environment closely replicated the setup of prior work [11, 21], ensuring consistent visual encoding and interaction design (see Figure 1). All nodes were labelled with ascending numerical identifiers starting from 0. Task-relevant nodes were rendered in red (#FF0000), while task-irrelevant nodes appeared in yellow (#FFFF00) with a diameter of 0.01. All edges were rendered in white by the Line System function in Babylon.js.

Participants conducted the study within a VR environment that simulated a 5×5 metre room with dark blue walls. A virtual question-and-answer panel was positioned on one wall as the sole interaction interface. Graphs were rendered in the centre of the room at a height ranging from 0.5 to 1.5 metres, allowing participants to explore them freely from multiple perspectives. Participants interacted with the panel using the Meta Quest 3 controller through a raycasting-based technique. Before beginning the experiment, all participants received standardised training on this interaction method, ensuring consistent familiarity with the VR controls across sessions.

3.3.2 Experiment Protocol

Participants began the study by clicking a start button, after which a calibration phase was initiated following a 2-second delay. During calibration, participants were instructed to follow five white crosses presented sequentially on a blue background. Each cross was displayed for 1 second at the following locations on screen: centre, upper left, lower left, upper right, and lower right. A 1-second interval separated the disappearance of one cross and the appearance of the next. Following calibration, a graph-related question was shown on a question-answering panel. Participants had unlimited time to read the question. Upon clicking to begin the task, a graph visualisation appeared at the centre of the virtual room. Participants could freely inspect the graph and, once confident in their solution, return to the panel to click "Done", triggering a response screen. After submitting an answer, a confirmation screen allowed participants to revise their response if necessary. The next task appeared upon confirmation, and participants were not allowed to revisit previous tasks.

The study began with three tutorial tasks, each using a simple graph representing one of the three task types. The experimenter explained each task type and instructed participants to think aloud while solving them. Participants could proceed only if all tutorial tasks

[6] `https://immersiveweb.dev/`
[7] `https://www.babylonjs.com/`

were solved correctly. The remaining nine tasks were the actual study. Task types and graph stimuli were counterbalanced and rotated across participants to ensure that no two consecutive tasks were of the same type or used the same graph.

3.4 Hypothesis

Our study aims to understand the key factors influencing users' gaze behaviour in stereoscopic 3D and how this behaviour reflects answer correctness. As visual attention behaviour is influenced by both bottom-up (graph) and top-down (task) factors, we hypothesise the effects:

- H1: Graph significantly influences users' gaze behaviour during graph perception in stereoscopic 3D.
- H2: Task type significantly influences users' gaze behaviour during graph perception in stereoscopic 3D.
- H3: Users' gaze behaviour is significantly correlated with answer correctness during graph perception in stereoscopic 3D.

We break the hypotheses down under these gaze metrics: a) saccade velocity, b) saccade length, and c) the ratio of fixation count landing on task-related elements. As such, every hypothesis consists of three sub-hypotheses, such as H1.b, which states that the graph significantly affects saccade length during graph perception in stereoscopic 3D.

4 Result

The ethics committee at the local university approved the user study. We conducted our user study at the local university. We recruited 18 participants (11 male, 7 female) between 21 and 28 years old ($\mu = 25.2$, $\sigma = 2$ years). All participants had no visual impairment but limited experience with VR. All 18 participants were counterbalanced across the three graphs and three tasks, resulting in nine perceptual tasks for each participant. Participants were compensated with 10€. The headset setup and eye tracker calibration lasted about five minutes, and the tutorial tasks took each participant about three minutes.

4.1 Data Processing

The image plane of the scene camera measured 1,600×1,200 pixels, covering a 103°×77° field of view [1]. We logged the VR headset's position and rotation with timestamps and event markers (e.g., task start/done) to segment the collected gaze data. The eye tracker recorded gaze direction as azimuth and elevation at 200 Hz. Fixations were identified using an enhanced Identify by Velocity Threshold (I-VT) algorithm for head-mounted eye tracking [9].

We synchronised timestamps between the VR system and the eye tracker using data from the initial calibration phase (see Section 3.3.2). The five-point fixation pattern was extracted from the eye tracking data and aligned with system log timestamps marking when each calibration cross was rendered. The average time difference across all five points was used to compute a temporal offset between the two systems.

To analyse participants' perceptual engagement with the graph visualisation, we reconstructed the full virtual environment using the Python Trimesh library[8]. Nodes and edges in the graph were rendered at 5 times their actual size used in the user study environment for

[8] https://trimesh.org/

ray-mesh intersections. Ray casting was then employed to identify graph elements intersected by participants' gaze. Gaze rays were computed by combining eye tracking data with the VR headset's position and orientation. Each ray originated from the headset's position, with its direction derived from the headset's rotation and the eye tracker's azimuth and elevation. The headset's quaternion rotation $q = (x, y, z, w)$ is converted into a rotation matrix by:

$$\mathbf{R}_{VR} = \begin{pmatrix} 1-2(y^2+z^2) & 2(xy-wz) & 2(xz+wy) \\ 2(xy+wz) & 1-2(x^2+z^2) & 2(yz-wx) \\ 2(xz-wy) & 2(yz+wx) & 1-2(x^2+y^2) \end{pmatrix} \tag{1}$$

The eye tracker provided fixation data in terms of azimuth (θ_{az}) and elevation (θ_{el}) angles, which were converted from degrees to radians. To compute the gaze direction vector, we defined the initial forward direction in the local VR headset space as $\vec{v}_{local} = (0, 0, 1)$. Rotation matrices for the elevation (pitch) and azimuth (yaw) angles were then constructed:

$$\mathbf{R}_{el} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_{el}) & -\sin(\theta_{el}) \\ 0 & \sin(\theta_{el}) & \cos(\theta_{el}) \end{pmatrix} \tag{2}$$

$$\mathbf{R}_{az} = \begin{pmatrix} \cos(\theta_{az}) & 0 & \sin(\theta_{az}) \\ 0 & 1 & 0 \\ -\sin(\theta_{az}) & 0 & \cos(\theta_{az}) \end{pmatrix} \tag{3}$$

The local gaze direction was then calculated by applying these rotations to the forward vector $\vec{v}_{gaze_local} = \mathbf{R}_{az}\mathbf{R}_{el}\vec{v}_{local}$. Finally, this local direction was transformed into world coordinates using the VR headset's rotation matrix and normalised to obtain the central ray direction $\vec{d}_{ray} = \frac{\mathbf{R}_{VR}\vec{v}_{gaze_local}}{|\mathbf{R}_{VR}\vec{v}_{gaze_local}|}$.

A single gaze ray is often insufficient for reliable object intersection due to eye tracking inaccuracies and the spatial extent of foveal vision [28, 34]. Prior work has shown that cone-based approaches with visual angles between 3–10° enhance robustness to head movement and positioning variability, while maintaining accurate detection in 3D environments [20].

Following these guidelines, we expanded the central gaze ray into a conical projection with an angle of $\alpha = 3°$. To generate the cone, we constructed an orthonormal basis around the central gaze direction $\vec{d_c}$ by identifying two perpendicular unit vectors $\vec{v}_1$ and $\vec{v}_2$, such that $\vec{d_c}, \vec{v}_1, \vec{v}_2$ formed an orthonormal frame. Eight surrounding ray directions were then computed as $\vec{d_i} = \cos\alpha\vec{d_c} + \sin\alpha(\cos\theta_i\vec{v}_1 + \sin\theta_i\vec{v}_2)$, where $\theta_i = \frac{2\pi i}{n}$, $i \in 0, 1, ..., n-1$, and $n = 8$. All directions $\vec{d_i}$ were normalised to unit length. Ray casting was performed using this nine-ray cone (the central ray plus eight surrounding rays) within the reconstructed 3D graph environment. All intersected graph elements and their respective distances from the ray origin were recorded.

4.2 Taxonomy of Task-related Graph Elements

Graph elements (meshes) were systematically categorised into eight categories: question panel, highlighted nodes, task nodes, task edges, confusion nodes, confusion edges, other nodes, and other edges.

Highlighted nodes are the nodes colour-coded in red, which are two nodes in the shortest path (T_{SP}) and identify common neighbours task (T_{CN}), but the highest degree task (T_{HD}) contains no highlighted nodes.

Task-relevant elements vary according to the specific task conditions:

- T_{SP}: The nodes and edges compose the shortest path(s) between the two highlighted nodes. When multiple shortest paths exist, all constituent elements are included.
- T_{HD}: The nodes include the node with the highest degree, and the edges include all edges connecting to these highest-degree nodes.
- T_{CN}: The nodes represent the common neighbours shared by both highlighted nodes, and the edges include all connections between these common neighbours and the highlighted nodes.

Confusion elements represent potentially misleading graph components that could interfere with task completion:

- T_{SP}: Confusion nodes and edges comprise all nodes and edges belonging to the second shortest path(s) between the highlighted nodes. When multiple second-shortest paths exist, all constituent elements are included.
- T_{HD}: Confusion nodes include all nodes with the second-highest degree, while confusion edges encompass their connections. When multiple nodes share the second-highest degree, all such nodes and their edges are classified as confusion elements.
- T_{CN}: Confusion nodes encompass all neighbours of either highlighted node (including both common and non-common neighbours), with confusion edges representing all connections between these neighbours and the highlighted nodes.

Other elements include all remaining nodes and edges that do not fall into the above categories. When a graph element could be assigned to multiple categories, we applied the following hierarchical priority: highlighted nodes > task nodes/edges > confusion nodes/edges > other elements.

4.3 Result

4.3.1 Task Performance Overview

The average task completion time is 53.85 seconds ($\sigma = 33.28$ s), where Graph_{sy}, Graph_{ic}, and Graph_{ed} are 57.44 ($\sigma = 34.00$ s), 69.20 ($\sigma = 36.35$ s), and 34.93 ($\sigma = 16.41$ s) seconds, respectively. For tasks T_{SP}, T_{HD}, and T_{CN}, participants spent 50.55 ($\sigma = 31.29$ s), 57.45 ($\sigma = 33.38$ s), and 53.56 ($\sigma = 35.30$ s) seconds, respectively. Participants have an average answer correctness of 78.40%, while T_{SP}, T_{HD}, and T_{CN} have a correctness of 79.63%, 88.89%, and 66.67%, respectively. The correctness of the three graphs is 83.33%, 62.96%, 88.89%, respectively. See Figure 2 for the average task correctness and duration across task types and graphs.

4.3.2 Gaze Behaviour

As shown in Figure 3, participants have a mean saccade velocity of 27.28 °/seconds ($\sigma = 5.00°$), and a mean saccade length of 7.89° ($\sigma = 1.45°$). For saccade velocity, a two-way ANOVA revealed that graph has a significant influence, $F(\text{graph}) = 9.201$, $p < 0.001$. However, the impact of task types was insignificant, $F(\text{task}) = 1.601$, $p = 0.205$, interaction effect $F(\text{graph:task}) = 1.613$, $p = 0.174$. The hypothesis H1.a is confirmed, but H2.a is rejected. For saccade length, a two-way ANOVA revealed that graph has a significant influence, $F(\text{graph}) = 4.553$, $p = 0.012$. However, the impact of task types was insignificant, $F(\text{task}) = 1.210$, $p = 0.301$, interaction effect $F(\text{graph:task}) = 1.611$, $p = 0.174$. The hypothesis H1.b is confirmed, but H2.b is rejected.

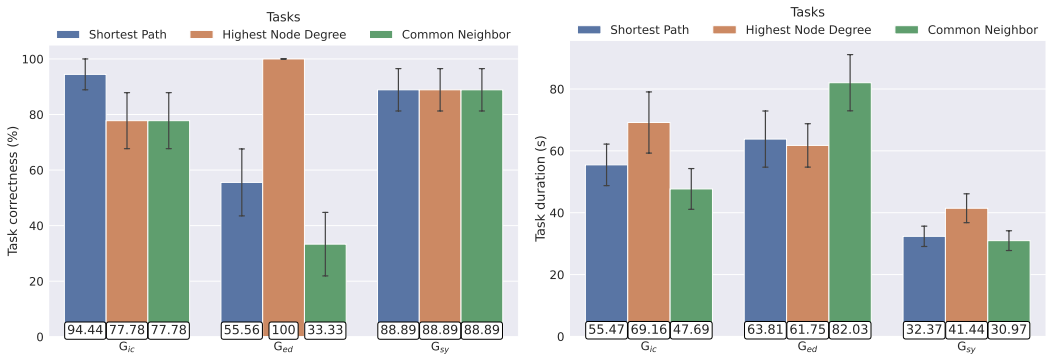


Figure 2 Average task correctness and duration across task types and graphs. Error bars indicate the standard error.

Participants have a mean fixation count of 145.65 ($\sigma = 84.95$), where 149.11 ($\sigma = 94.11$) for T_{SP}, 139.61 ($\sigma = 75.26$) for T_{HD}, and 148.24 ($\sigma = 85.71$) for T_{CN}. Separating by graph, the means were 154.19 ($\sigma = 82.83$) for $Graph_{sy}$, 181.98 ($\sigma = 96.65$) for $Graph_{ic}$, and 100.80 ($\sigma = 47.45$) for $Graph_{ed}$. Considering that $Graph_{ic}$ has a significantly larger number of fixation counts than $Graph_{sy}$ and $Graph_{ed}$, we used the task-related fixation ratio for the following tests. For task-related fixation ratio, a two-way ANOVA revealed that both graph and task have a significant influence, $F(\text{graph}) = 22.758$, $p < 0.001$, $F(\text{task}) = 36.233$, $p < 0.001$, interaction effect $F(\text{graph:task}) = 7.063$, $p < 0.001$. A post-hoc Tukey Test confirmed the pairwise significant differences between T_{SP} and T_{CN} in $Graph_{sy}$ and $Graph_{ic}$, and between T_{SP} and T_{HD}, T_{SP} and T_{HD} in $Graph_{ed}$ (see Figure 3). This confirmed hypotheses H1.c and H2.c.

For T_{SP}, the correlation between answer correctness with saccade velocity, saccade length, and fixation task ratio are all insignificant, Spearman's Rank Correlation $r_s = 0.0310$, $p = 0.824$, $r_s = 0.0634$, $p = 0.649$, and, $r_s = -0.010$, $p = 0.941$, respectively. Answer correctness is significantly correlated with saccade velocity, saccade length, and fixation task ratio under T_{CN}, Spearman's Rank Correlation $r_s = 0.4008$, $p = 0.003$, $r_s = 0.3856$, $p = 0.004$, $r_s = -0.3529$, $p = 0.009$, respectively. For T_{HD}, the correlation between answer correctness with saccade velocity, and fixation task ratio are insignificant, Spearman's Rank Correlation $r_s = 0.2363$, $p = 0.085$, $r_s = 0.1853$, $p = 0.180$, respectively, but significantly correlated to saccade length, $r_s = 0.2741$, $p = 0.045$. This leads to the confirmation of H3.a, H3.b, and H3.c under T_{CN}, but rejection in T_{SP} and T_{HD}. Moreover, under T_{CN}, answer correctness correlates significantly negatively with task duration, Spearman's Rank Correlation $r_s = -0.4915$, $p < 0.001$. This suggests that some participants spent a huge effort in counting common neighbours, but still answered incorrectly.

4.3.3 Viewing Strategy

We adapted two effective visualisation techniques to have both temporal and spatial representation of how users observe graphs under task solving, providing qualitative insights into the users' strategies under immersive graph navigation.

Temporal Dynamics

Implementation. To capture the temporal dynamics of participants' gaze behaviour, we implemented a timeline visualisation in the form of uncertainty-aware scarf plots [37]. In this visualisation, each vertical bar represents a single fixation event, with bar width encoding the

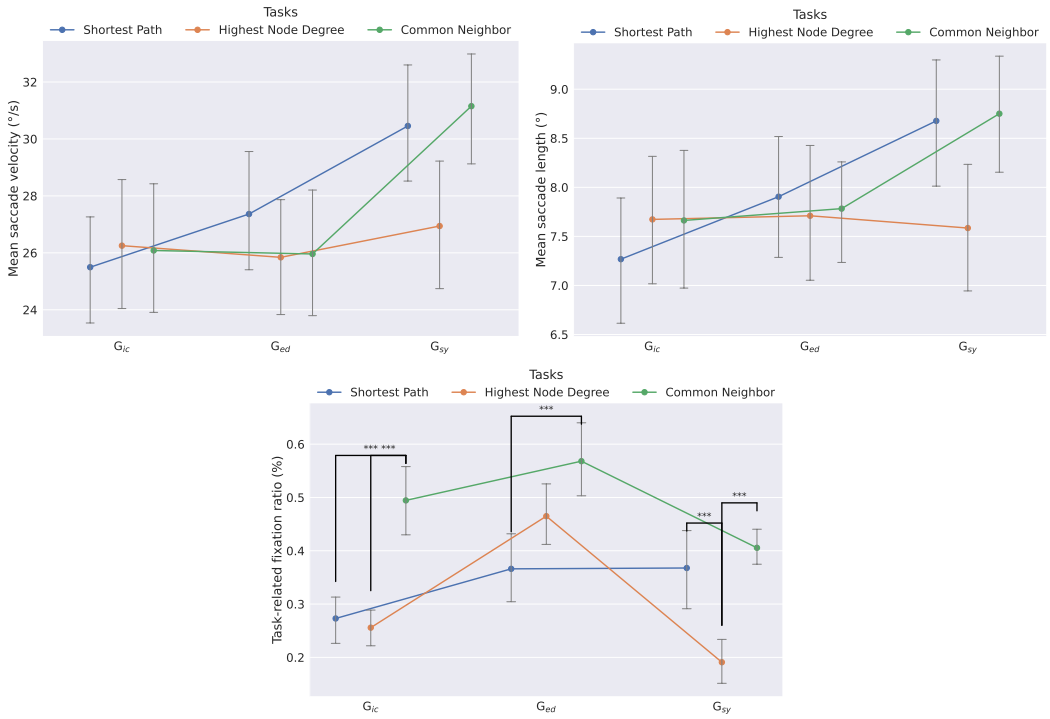


Figure 3 We compared task-related fixation ratio, mean saccade length, and mean saccade velocity across graphs and tasks. *** indicates $p < 0.001$. Error bars denote 95% confidence intervals.

fixation duration. Within each bar, coloured segments indicate the graph elements (nodes and edges) intersected by the participant's gaze ray during that fixation. For each intersected mesh i within a fixation, we calculated a weight w_i based on the distance d_i between the mesh and the participant's viewpoint:

$$w_i = \begin{cases} 0 & \text{if } d_i = 0 \\ \frac{1}{d_i} & \text{if } d_i \neq 0 \end{cases} \tag{4}$$

The proportion of vertical space allocated to each mesh block within the fixation bar was then determined by normalising these weights by $p_i = \frac{w_i}{\sum_j w_j}$, where p_i represents the height proportion of block i relative to the total bar height. Meshes closer to the observer (smaller d_i) receive proportionally larger block heights and are positioned lower within the bar. The colour coding corresponds to our eight-category classification described in Section 4.1.

Finding. The scarf plot (see Figure 4) revealed several noteworthy patterns. First, the appearance of task-related meshes usually appeared later than highlighted nodes and confusion meshes. Second, the number of intersected meshes generally decreased over time: the peak of mesh hit count usually appeared early during viewing (first 15 seconds), then gradually reduced to four or fewer meshes when attending to task-relevant meshes. Third, task-related meshes were usually first located farther away from viewers (top), then came closer to the foreground. This suggests that participants would adjust their viewpoints to reduce visual clutter and minimise the spatial distance to relevant mesh elements, thereby improving task-relevant visibility.

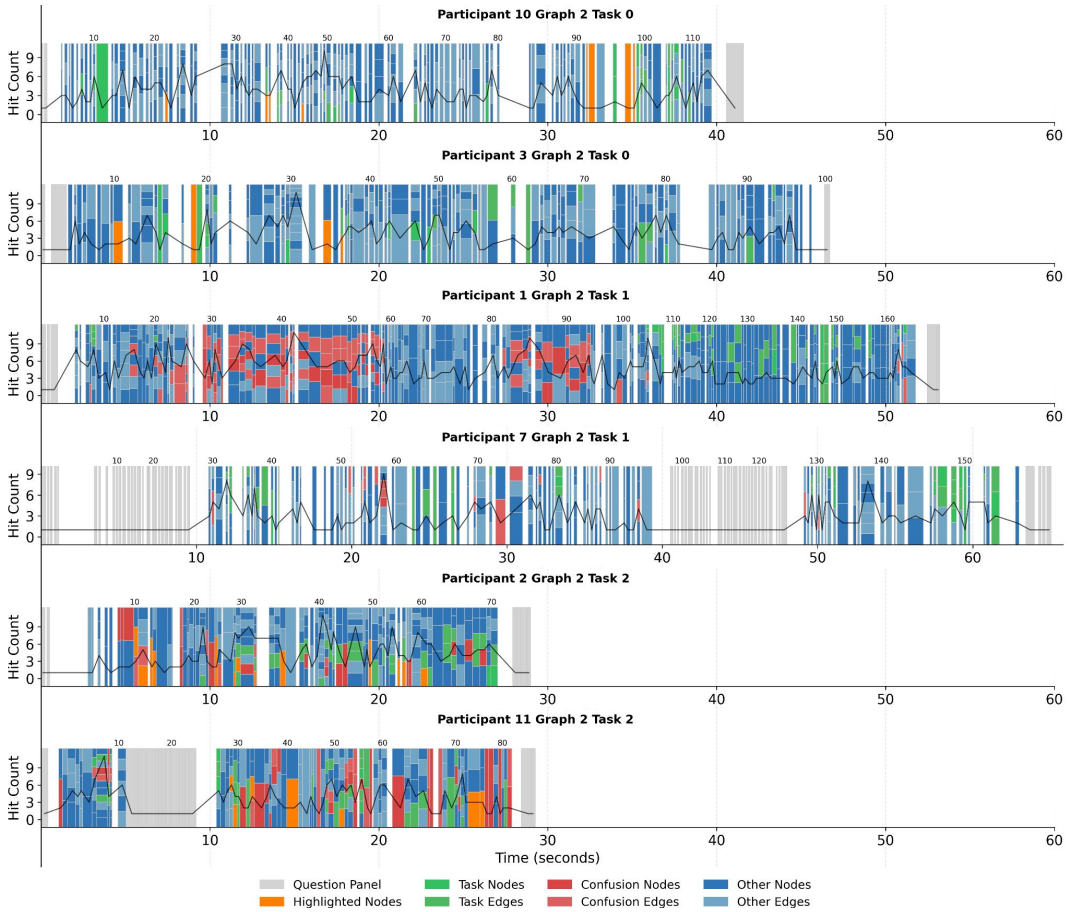


Figure 4 Sample timeline visualisations. Each vertical bar represents a fixation event, with width indicating duration and coloured blocks showing graph elements that intersected with the eye gaze ray. In each block, objects are displayed from closest (bottom) to farthest (top).

Spatial Distribution

Implementation. To understand the distribution of task-based visual attention, we first overlaid visual saliency values on graph meshes. For each fixation that intersected n mesh elements, we incremented the hit count of each intersected mesh by $\frac{1}{n}$. The resulting hit counts were mapped to a colour scale through linear interpolation. Given a mesh with hit count v, minimum hit count v_{min}, and maximum hit count v_{max}, we first computed the normalized value:

$$\hat{v} = \frac{\min(v, v_{max}) - v_{min}}{v_{max} - v_{min}}. \tag{5}$$

We then applied gamma correction with the gamma value of 0.7 to emphasise differences in the lower range, and the scaling factor of 1.2:

$$\hat{v}_{transformed} = \text{clip}(1.2(\hat{v})^{0.7}, 0, 1). \tag{6}$$

The RGB colour was then calculated using linear interpolation between white $\mathbf{C}_{white} = (255, 255, 255)$ and dark blue $\mathbf{C}_{blue} = (0, 0, 80)$, and then clipped to the range $[0, 255]$:

$$\mathbf{C} = \mathbf{C}_{white} + \hat{v}_{transformed} \cdot (\mathbf{C}_{blue} - \mathbf{C}_{white}). \tag{7}$$

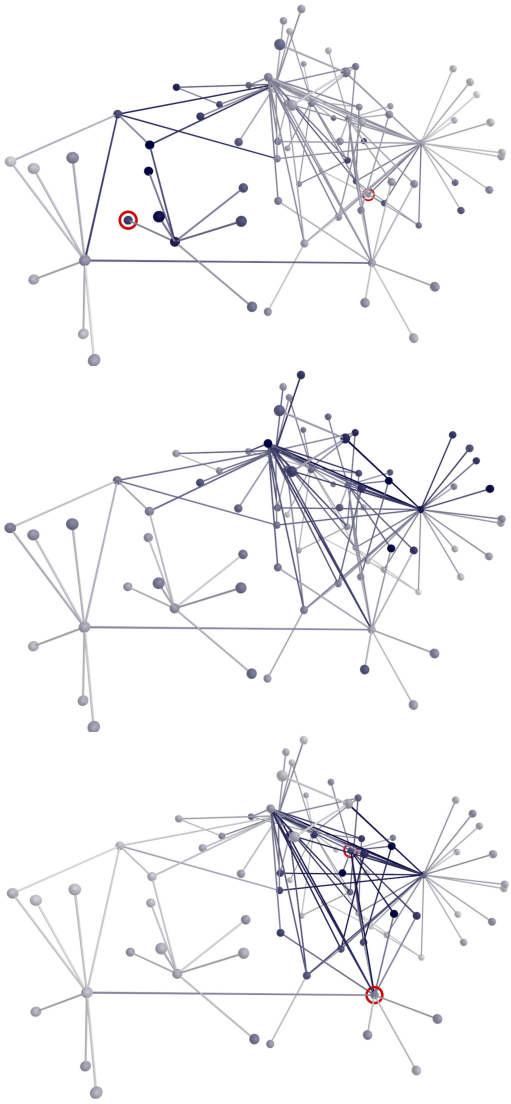


Figure 5 Aggregated visual saliency distribution from all participants of Graph$_{ic}$, where red circles denote the highlighted nodes. **Top**: task shortest path. **Middle**: task highest node degree. **Bottom**: task common neighbour.

Finding. We aggregated fixation data from all participants within each task-graph combination, computing hit frequencies for individual mesh elements throughout the graph (see Figure 5). The resulting saliency distribution revealed that visual attention allocation emerged from an interplay between task demands and graph structure. Participants adapted their visual strategies to task requirements while being guided by the graph's topological features, creating distinct yet structured attention patterns. The concentration of fixations on areas with many connections and relationships – such as densely connected nodes, nodes linking different parts of the graph, and tightly interconnected groups – indicates that viewers leveraged the graph's inherent organisation to efficiently locate task-relevant information, rather than employing exhaustive search strategies. This dual influence of task constraints and graph topology demonstrates that human visual processing of graphs involves a dynamic optimisation between goal-directed search and structure-guided exploration.

5 Discussion

5.1 Graphs, Tasks, and Gaze Behaviour

Eye tracking offers an objective framework for assessing perceptual and cognitive engagement during graph viewing. Our findings support the hypothesis regarding graph influence (H1), while revealing more nuanced effects related to task type and answer correctness (H2, H3).

Saccadic measures strongly support graph-related hypotheses. Different graphs significantly influenced both saccade velocity (H1.a) and saccade length (H1.b), whereas task type and correctness had no significant effects (rejecting H2.a, H2.b, H3.a, and H3.b). The data reveal a systematic inverse relationship between graph density and saccadic behaviour. Graph density varied across graphs: Graph_{sy} (4.0 degree), Graph_{ic} (3.12 degree), and Graph_{ed} (2.32 degree). Correspondingly, saccadic parameters showed systematic progression: lengths of 7.53°, 7.80°, and 8.34°, and velocities of 25.94°/s, 26.39°/s, and 29.52°/s, respectively. This monotonic pattern demonstrates that increasing graph density leads to more constrained visual scanning in localised regions, whilst sparser structures promote broader exploration. These distinct saccadic strategies likely reflect underlying cognitive demands, highlighting the utility of eye movement metrics as physiological indicators of visual processing in graph analysis.

Task-related fixation ratio analyses further support all three hypotheses regarding attention allocation (H1.c, H2.c, H3.c). Different graphs had multiplicative effects on attention demands: Graph_{ic}, with the highest structural complexity (75 nodes, 114 edges), elicited the greatest task-relevant fixation ratio (46.64%). Graph_{sy} (50 nodes, 100 edges) and Graph_{ed} (56 nodes, 65 edges) required substantially less attention to task-relevant elements (34.11% and 32.13%, respectively). Task type also shaped attentional demands: the common neighbours task (T_{CN}) imposed the greatest cognitive load, with 66.67% task correctness [30], compared to 79.63% for the shortest path (T_{SP}) and 88.89% for the highest node degree (T_{HD}) tasks. Finally, attention allocation correlated with correctness: the most complex graph (Graph_{ic}) and the most demanding task (T_{CN}) yielded the lowest task correctness (33.33%).

5.2 Spatio-temporal Attention Patterns in Graph Perception

Eye tracking captures rich spatio-temporal data, enabling sophisticated visualisation of visual attention patterns. The temporal precision and spatial accuracy of gaze data support advanced analytical techniques, including scarf plots that reveal the temporal evolution of participants' focus and the sequential ordering of graph element inspection, and saliency distribution that characterise the distribution of human visual attention across graph structures.

Our visualisation analysis reveals distinct task-specific attention strategies that reflect underlying cognitive processes. In the shortest path task, participants exhibited distributed fixation patterns across intermediate vertices, consistent with systematic route exploration. Saliency analysis (Figure 5) showed moderate activation levels among candidate nodes, while scarf plots (Figure 4) revealed frequent transitions between task-relevant and irrelevant vertices. These patterns suggest a breadth-first visual search strategy, where participants actively compared multiple paths before selecting an optimal route. In contrast, the highest node degree task prompted more focused visual behaviour. Visual salience was concentrated on structurally prominent nodes – appearing as dark blue clusters in the saliency distribution – while scarf plots showed fewer transitions between relevant and irrelevant areas. This indicates a targeted search strategy, with participants quickly identifying high-degree vertices with minimal exploration. The common neighbours task exhibited a distinctly localised

attention pattern. Participants primarily fixated on the neighbourhoods of the highlighted nodes, as shown by tightly clustered saliency regions and infrequent transitions in scarf plots. This behaviour reflects a local comparison strategy, where participants systematically inspected nearby nodes to infer shared connections, rather than scanning the broader graph.

Despite these task-specific strategies, some consistent visual attention patterns emerged. In particular, the hub nodes attracted attention across all tasks, implying that participants consistently prioritised structurally significant elements during their analysis.

5.3 Limitations and Future Work

Challenges of Perception Analysis in Graph Visualisation

Analysing visual attention patterns in 3D graph visualisations presents substantial methodological constraints. The primary limitation arises from depth ambiguity, where multiple mesh elements at varying depths align along identical sight vectors. Our conical ray casting detects all intersected meshes, creating uncertainty regarding actual fixation targets. This ambiguity intensifies in dense regions where individual fixations often intersect over ten elements, most likely outside the participant's attentional focus. Although restricting detection to the nearest mesh offers a straightforward solution, this risks misclassification when participants attend to background elements.

A secondary limitation emerges from long-distance edges spanning extensive 3D regions. Current protocols classify entire edges as fixation targets upon any segment intersection, generating spurious detections and artificially inflating attention frequencies. Spatial partitioning into discrete subregions offers a potential solution by recording intersections at regional rather than mesh levels, though this approach requires accurate gaze depth computation [22] to determine the specific attended subregion.

Real-time Gaze-based Applications

Our study was constrained by technical limitations related to the integration of eye tracking and VR system. Specifically, fixation data were computed offline from raw gaze recordings, and no real-time communication occurred between the eye tracker and the HMD. This restricted our ability to support gaze-contingent interactions, such as highlighting graph elements under direct visual attention or dynamically adapting the graph layout based on live eye tracking metrics. These types of interactions have been shown to enhance usability and user engagement in immersive analytics systems. Future work could address this limitation by implementing a web socket to enable real-time data transmission between the eye tracker and the VR application.

Additionally, the asynchronous nature of the two systems necessitated a post hoc timestamp alignment process, based on a five-point validation. While generally effective, this method may introduce minor misalignments – typically on the order of a few milliseconds – due to human reaction time variability. Although such discrepancies are unlikely to affect high-level attention analyses, they may pose challenges for fine-grained temporal studies or real-time applications. Addressing this synchronisation issue remains an important technical challenge for future system designs in immersive eye tracking research.

6 Conclusion

This work contributes a comprehensive eye tracking study for understanding human perception of graph visualisations in immersive VR environments through visual analytics. With the spatial-temporal gaze pattern analysis, our 18-participant user study reveals systematic

relationships between graph analytical tasks and corresponding gaze behaviours, demonstrating that eye tracking metrics can serve as objective measures of cognitive load and visual complexity in exploring 3D graph visualisation. These results provide foundational insights for designing more effective immersive visualisation systems and suggest promising directions for adaptive interfaces that respond to user cognitive states. Future work should explore the generalisability of these gaze-based indicators across more graphs and tasks, as well as investigate real-time applications of these metrics for enhanced user experience in immersive analytics.

References

1 Chris Baumann and Kai Dierkes. Neon accuracy test report. *Pupil Labs*, 10, 2023.

2 D. Belcher, M. Billinghurst, S.E. Hayes, and R. Stiles. Using augmented reality for visualizing complex graphs in three dimensions. In *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, pages 84–93. IEEE, 2003. doi:10.1109/ISMAR.2003.1240691.

3 T. Blascheck, K. Kurzhals, M. Raschke, M. Burch, D. Weiskopf, and T. Ertl. State-of-the-Art of Visualization for Eye Tracking Data . In R. Borgo, R. Maciejewski, and I. Viola, editors, *EuroVis - STARs*. The Eurographics Association, 2014. doi:10.2312/eurovisstar.20141173.

4 Wolfgang Büschel, Jian Chen, Raimund Dachselt, Steven Drucker, Tim Dwyer, Carsten Görg, Tobias Isenberg, Andreas Kerren, Chris North, and Wolfgang Stuerzlinger. Interaction for immersive analytics. *Immersive analytics*, pages 95–138, 2018. doi:10.1007/978-3-030-01388-2_4.

5 Wolfgang Büschel, Stefan Vogt, and Raimund Dachselt. Augmented reality graph visualizations. *IEEE Computer Graphics and Applications*, 39(3):29–40, 2019. doi:10.1109/MCG.2019.2897927.

6 Kun-Ting Chen, Quynh Quang Ngo, Kuno Kurzhals, Kim Marriott, Tim Dwyer, Michael Sedlmair, and Daniel Weiskopf. Reading strategies for graph visualizations that wrap around in torus topology. In *Proceedings of the 2023 Symposium on Eye Tracking Research and Applications*, ETRA '23, pages 1–7, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3588015.3589841.

7 Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W. Klau, Karsten Klein, and Petra Mutzel. The open graph drawing framework (ogdf). *Handbook of graph drawing and visualization*, 2011:543–569, 2013.

8 Giorgio Conte, Allen Q. Ye, Angus G. Forbes, Olusola Ajilore, and Alex Leow. Braintrinsic: A virtual reality-compatible tool for exploring intrinsic topologies of the human brain connectome. In *Brain Informatics and Health*, pages 67–76, Cham, 2015. Springer International Publishing. doi:10.1007/978-3-319-23344-4_7.

9 Michael Drews and Kai Dierkes. Strategies for enhancing automatic fixation detection in head-mounted eye tracking. *Behavior Research Methods*, 56(6):6276–6298, 2024. doi:10.3758/s13428-024-02360-0.

10 Adam Drogemuller, Andrew Cunningham, James Walsh, Bruce H Thomas, Maxime Cordeil, and William Ross. Examining virtual reality navigation techniques for 3d network visualisations. *Journal of Computer Languages*, 56:100937, 2020. doi:10.1016/j.cola.2019.100937.

11 Stefan P Feyer, Bruno Pinaud, Stephen Kobourov, Nicolas Brich, Michael Krone, Andreas Kerren, Michael Behrisch, Falk Schreiber, and Karsten Klein. 2d, 2.5 d, or 3d? an exploratory study on multilayer network visualisations in virtual reality. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):469–479, 2023. doi:10.1109/TVCG.2023.3327402.

12 Adrien Fonnet and Yannick Prié. Survey of immersive analytics. *IEEE Transactions on Visualization and Computer Graphics*, 27(3):2101–2122, 2021. doi:10.1109/TVCG.2019.2929033.

13 Nicolas Greffard, Fabien Picarougne, and Pascale Kuntz. Visual community detection: An evaluation of 2d, 3d perspective and 3d stereoscopic displays. In *Graph Drawing*, pages 215–225, Berlin, Heidelberg, 2012. Springer. doi:10.1007/978-3-642-25878-7_21.

14 Nicolas Greffard, Fabien Picarougne, and Pascale Kuntz. Beyond the classical monoscopic 3D in graph analytics: An experimental study of the impact of stereoscopy. In *2014 IEEE VIS International Workshop on 3DVis (3DVis)*, pages 19–24, 2014. doi:10.1109/3DVis.2014.7160095.

15 Weidong Huang. Using eye tracking to investigate graph layout effects. In *2007 6th International Asia-Pacific Symposium on Visualization*, pages 97–100. IEEE, 2007. doi:10.1109/APVIS.2007.329282.

16 Weidong Huang. Establishing aesthetics based on human graph reading behavior: two eye tracking studies. *Personal and ubiquitous computing*, 17:93–105, 2013. doi:10.1007/s00779-011-0473-2.

17 Sabrina Jaeger, Karsten Klein, Lucas Joos, Johannes Zagermann, Michael De Ridder, Jinman Kim, Jean Yang, Ulrike Pfeil, Harald Reiterer, and Falk Schreiber. Challenges for brain data analysis in VR environments. In *2019 IEEE Pacific Visualization Symposium (PacificVis)*, pages 42–46. IEEE, 2019. doi:10.1109/PacificVis.2019.00013.

18 Lucas Joos, Maximilian T. Fischer, Daniel A. Keim, and Johannes Fuchs. Aesthetic-driven navigation for node-link diagrams in vr. In *Proceedings of the 2023 ACM Symposium on Spatial User Interaction*, SUI '23, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3607822.3614537.

19 Lucas Joos, Maximilian T Fischer, Julius Rauscher, Daniel A Keim, Tim Dwyer, Falk Schreiber, and Karsten Klein. Visual network analysis in immersive environments: A survey. *arXiv preprint arXiv:2501.08500*, 2025. doi:10.48550/arXiv.2501.08500.

20 Anuradha Kar and Peter Corcoran. Performance evaluation strategies for eye gaze estimation systems with quantitative metrics and visualizations. *Sensors*, 18(9), 2018. doi:10.3390/s18093151.

21 Wilhelm Kerle-Malcharek, Stefan Paul Feyer, Falk Schreiber, and Karsten Klein. GAV-VR: An Extensible Framework for Graph Analysis and Visualisation in Virtual Reality. In Jean-Marie Normand, Maki Sugimoto, and Veronica Sundstedt, editors, *ICAT-EGVE 2023 - International Conference on Artificial Reality and Telexistence and Eurographics Symposium on Virtual Environments*. The Eurographics Association, 2023. doi:10.2312/egve.20231321.

22 Maurice Koch, Nelusa Pathmanathan, Daniel Weiskopf, and Kuno Kurzhals. How deep is your gaze? leveraging distance in image-based gaze analysis. In *Proceedings of the 2024 Symposium on Eye Tracking Research and Applications*, pages 1–7, 2024. doi:10.1145/3649902.3653349.

23 Matthias Kraus, Johannes Fuchs, Björn Sommer, Karsten Klein, Ulrich Engelke, Daniel Keim, and Falk Schreiber. Immersive analytics with abstract 3d visualizations: A survey. In *Computer Graphics Forum*, volume 41, pages 201–229. Wiley Online Library, 2022. doi:10.1111/cgf.14430.

24 Marcel Krüger, Qin Li, Torsten W. Kuhlen, and Tim Gerrits. A case study on providing immersive visualization for neuronal network data using cots soft- and hardware. In *2023 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 201–205, 2023. doi:10.1109/VRW58643.2023.00050.

25 Oh-Hyun Kwon, Chris Muelder, Kyungwon Lee, and Kwan-Liu Ma. A study of layout, rendering, and interaction methods for immersive graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 22(7):1802–1815, 2016. doi:10.1109/TVCG.2016.2520921.

26 Elektra Kypridemou, Michele Zito, and Marco Bertamini. The effect of graph layout on the perception of graph properties. *EuroVis (Short Papers)*, pages 1–5, 2020. doi:10.2312/evs.20201039.

27 Christof Körner. Sequential processing in comprehension of hierarchical graphs. *Applied Cognitive Psychology*, 18(4):467–480, 2004. doi:10.1002/acp.997.

28 Ruian Liu, Shijiu Jin, and Xiaorong Wu. Single camera remote eye gaze tracking under natural head movements. In *Advances in Artificial Reality and Tele-Existence*, pages 614–623, Berlin, Heidelberg, 2006. Springer. doi:10.1007/11941354_6.

29 Ana Beatriz Marques, Vasco Branco, Rui Costa, and Nina Costa. Immersive unit visualization with augmented reality. *Multimodal Technologies and Interaction*, 7(10):98, 2023. doi:10.3390/mti7100098.

30 AP Martinez-Cedillo, Nicoleta Gavrila, Aastha Mishra, Elena Geangu, and Tom Foulsham. Cognitive load affects gaze dynamics during real-world tasks. *Experimental Brain Research*, 243(4):82, 2025.

31 Fintan McGee, Benjamin Renoust, Daniel Archambault, Mohammad Ghoniem, Andreas Kerren, Bruno Pinaud, Margit Pohl, Benoît Otjacques, Guy Melançon, and Tatiana Von Landesberger. *Visual analysis of multilayer networks*, volume 8. Springer, 2021.

32 Jacob Miller, Mohammad Ghoniem, Hsiang-Yun Wu, and Helen C Purchase. On the perception of small sub-graphs. In *International Symposium on Graph Drawing and Network Visualization*, pages 213–230. Springer, Springer, 2023. doi:10.1007/978-3-031-49272-3.

33 Gavin J. Mooney, Helen C. Purchase, Michael Wybrow, Stephen G. Kobourov, and Jacob Miller. The Perception of Stress in Graph Drawings. In Stefan Felsner and Karsten Klein, editors, *32nd International Symposium on Graph Drawing and Network Visualization (GD 2024)*, volume 320 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:17, 2024. doi:10.4230/LIPIcs.GD.2024.21.

34 Fabricio Batista Narcizo, Fernando Eustáquio Dantas dos Santos, and Dan Witzner Hansen. High-accuracy gaze estimation for interpolation-based eye-tracking methods. *Vision*, 5(3), 2021. doi:10.3390/vision5030041.

35 Rudolf Netzel, Michel Burch, and Daniel Weiskopf. Comparative eye tracking study on node-link visualizations of trajectories. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2221–2230, 2014. doi:10.1109/TVCG.2014.2346420.

36 Mershack Okoe, Sayeed Safayet Alam, and Radu Jianu. A gaze-enabled graph visualization to improve graph reading tasks. *Computer Graphics Forum*, 33(3):251–260, 2014. doi:10.1111/cgf.12381.

37 Nelusa Pathmanathan, Seyda Öney, Maurice Koch, Daniel Weiskopf, and Kuno Kurzhals. Uncertainty-aware scarf plots. In *Proceedings of the 2025 Symposium on Eye Tracking Research and Applications*, pages 1–8. Association for Computing Machinery, 2025. doi:10.1145/3715669.3725872.

38 Alexander Plopski, Teresa Hirzle, Nahal Norouzi, Long Qian, Gerd Bruder, and Tobias Langlotz. The eye in extended reality: A survey on gaze interaction and eye tracking in head-worn extended reality. *ACM Comput. Surv.*, 55(3), March 2022. doi:10.1145/3491207.

39 Mathias Pohl, Markus Schmitt, and Stephan Diehl. Comparing the Readability of Graph Layouts using Eyetracking and Task-oriented Analysis . In Oliver Deussen and Peter Hall, editors, *Computational Aesthetics in Graphics, Visualization, and Imaging*. The Eurographics Association, 2009. doi:10.2312/COMPAESTH/COMPAESTH09/049-056.

40 Bonita Sharif and Jonathan I. Maletic. An eye tracking study on the effects of layout in understanding the role of design patterns. In *2010 IEEE International Conference on Software Maintenance*, pages 1–10, 2010. doi:10.1109/ICSM.2010.5609582.

41 Utkarsh Soni, Yafeng Lu, Brett Hansen, Helen C. Purchase, Stephen G. Kobourov, and Ross Maciejewski. The perception of graph properties in graph layouts. *Comput. Graph. Forum*, 37(3):169–181, 2018. doi:10.1111/CGF.13410.

42 J. Sorger, A. Arleo, P. Kán, W. Knecht, and M. Waldner. Egocentric network exploration for immersive analytics. *Computer Graphics Forum*, 40(7):241–252, 2021. doi:10.1111/cgf.14417.

43 Ben Steichen, Giuseppe Carenini, and Cristina Conati. User-adaptive information visualization: using eye gaze data to infer visualization tasks and user cognitive abilities. In *Proceedings of*

the 2013 International Conference on Intelligent user interfaces, pages 317–328. Association for Computing Machinery, 2013. doi:10.1145/2449396.2449439.

44 Qi Sun, Budmonde Duinkharjav, and Anjul Patney. Modeling and optimizing human-in-the-loop visual perception using immersive displays: A review. In *SID symposium digest of technical papers*, volume 53, pages 190–193. Wiley Online Library, 2022. doi:10.1002/sdtp.15450.

45 Veronica Sundstedt and Valeria Garro. A systematic review of visualization techniques and analysis tools for eye-tracking in 3d environments. *Frontiers in Neuroergonomics*, 3, 2022. doi:10.3389/fnrgo.2022.910019.

46 M. Tory, M.S. Atkins, A.E. Kirkpatrick, M. Nicolaou, and G.-Z. Yang. Eyegaze analysis of displays with combined 2D and 3D views. In *VIS 05. IEEE Visualization, 2005.*, pages 519–526, 2005. doi:10.1109/VISUAL.2005.1532837.

47 Yao Wang, Yue Jiang, Zhiming Hu, Constantin Ruhdorfer, Mihai Bâce, and Andreas Bulling. VisRecall++: Analysing and predicting visualisation recallability from gaze behaviour. *Proceedings of the ACM on Human-Computer Interaction*, 8(ETRA):1–18, 2024. doi:10.1145/3655613.

48 Yao Wang, Weitian Wang, Abdullah Abdelhafez, Mayar Elfares, Zhiming Hu, Mihai Bâce, and Andreas Bulling. SalChartQA: Question-driven saliency on information visualisations. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–14. Association for Computing Machinery, 2024. doi:10.1145/3613904.3642942.

49 Colin Ware and Glenn Franck. Evaluating stereo and motion cues for visualizing information nets in three dimensions. *ACM Trans. Graph.*, 15(2):121–140, April 1996. doi:10.1145/234972.234975.

50 Colin Ware and Peter Mitchell. Reevaluating stereo and motion cues for visualizing graphs in three dimensions. In *Proceedings of the 2nd Symposium on Applied Perception in Graphics and Visualization*, APGV '05, pages 51–58, New York, NY, USA, August 2005. Association for Computing Machinery. doi:10.1145/1080402.1080411.

51 Colin Ware and Peter Mitchell. Visualizing graphs in three dimensions. *ACM Transactions on Applied Perception*, 5(1):1–15, January 2008. doi:10.1145/1279640.1279642.

52 Matt Whitlock, Stephen Smart, and Danielle Albers Szafir. Graphical perception for immersive analytics. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 616–625, 2020. doi:10.1109/VR46266.2020.00084.

53 Fumeng Yang, James Tompkin, Lane Harrison, and David H Laidlaw. Visual cue effects on a classification accuracy estimation task in immersive scatterplots. *IEEE Transactions on Visualization and Computer Graphics*, 29(12):4858–4873, 2022. doi:10.1109/TVCG.2022.3192364.

54 Vahan Yoghourdjian, Daniel Archambault, Stephan Diehl, Tim Dwyer, Karsten Klein, Helen C. Purchase, and Hsiang-Yun Wu. Exploring the limits of complexity: A survey of empirical studies on graph visualisation. *Visual Informatics*, 2(4):264–282, 2018. doi:10.1016/j.visinf.2018.12.006.

55 Vahan Yoghourdjian, Yalong Yang, Tim Dwyer, Lee Lawrence, Michael Wybrow, and Kim Marriott. Scalability of network visualisation from a cognitive load perspective. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1677–1687, 2020. doi:10.1109/TVCG.2020.3030459.

56 Xin Zhao, Shuowen Fu, Rui Yang, Lei Yang, Yunpeng Chen, Jiang Zhang, Jiang Long, Fangfang Zhou, and Ying Zhao. Investigating visual perception of degree centrality in graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–11, 2025. doi:10.1109/TVCG.2025.3567129.

Flipping Odd Matchings in Geometric and Combinatorial Settings

Oswin Aichholzer
Graz University of Technology, Austria

Sofia Brenner
University of Kassel, Germany

Joseph Dorfer
Graz University of Technology, Austria

Hung P. Hoang
TU Wien, Austria

Daniel Perz
University of Perugia, Italy

Christian Rieck
University of Kassel, Germany

Francesco Verciani
University of Kassel, Germany

Abstract

We study the problem of reconfiguring *odd matchings*, that is, matchings that cover all but a single vertex. Our reconfiguration operation is a so-called *flip* where the unmatched vertex of the first matching gets matched, while consequently another vertex becomes unmatched. We consider two distinct settings: the *geometric setting*, in which the vertices are points embedded in the plane and all occurring odd matchings are crossing-free, and a *combinatorial setting*, in which we consider odd matchings in general graphs.

For the latter setting, we provide a complete polynomial time checkable characterization of graphs in which any two odd matchings can be reconfigured into each another. This complements the previously known result that the flip graph is always connected in the geometric setting [2]. In the combinatorial setting, we prove that the diameter of the flip graph, if connected, is linear in the number of vertices. Furthermore, we establish that deciding whether there exists a flip sequence of length k transforming one given matching into another is NP-complete in both the combinatorial and the geometric settings. To prove the latter, we introduce a framework that allows us to transform partial order types into general position with only polynomial overhead. Finally, we demonstrate that when parameterized by the flip distance k, the problem is fixed-parameter tractable (FPT) in the geometric setting when restricted to convex point sets.

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

Keywords and phrases Odd matchings, reconfiguration, flip graph, geometric, combinatorial, connectivity, NP-hardness, FPT

Digital Object Identifier 10.4230/LIPIcs.GD.2025.12

Related Version *Full Version*: `https://arxiv.org/abs/2508.18457` [1]

Funding *Sofia Brenner*: German Research Foundation (DFG) grant 522790373.
Joseph Dorfer: Austrian Science Foundation (FWF) grant 10.55776/DOC183.
Hung P. Hoang: Austrian Science Foundation (FWF) project Y 1329 START-Programm.
Daniel Perz: MUR PRIN project 2022ME9Z78 - "NextGRAAL: Next-generation algorithms for constrained GRAph visuALization".
Christian Rieck: German Research Foundation (DFG) grant 522790373.
Francesco Verciani: German Research Foundation (DFG) grant 522790373.



© Oswin Aichholzer, Sofia Brenner, Joseph Dorfer, Hung P. Hoang, Daniel Perz, Christian Rieck, and Francesco Verciani;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

In many areas of discrete mathematics and theoretical computer science, one is interested not only in finding individual solutions to combinatorial problems, but also in understanding how these solutions relate to each other – specifically, whether it is possible to gradually transform one solution into another through a sequence of small, valid changes (so-called *flips*), while maintaining feasibility at each step. This framework is known as *reconfiguration*. It provides insight into the structure of the solution space and has applications in various areas. For example in optimization [21], as every optimization problem can be reformulated as a reconfiguration problem by introducing an appropriate threshold on the objective functions [18], as well as combinatorial enumeration and robustness [4, 22]. Moreover, many classical results in combinatorics naturally fall within this framework, e.g., the 5-color theorem [13] or Vizing's theorem [3, 27]. We also refer to the following surveys [23, 26]. A prominent example is the reconfiguration of triangulations of a convex n-gon; since each binary tree corresponds to the geometric dual of such a triangulation, this problem is equivalent to performing rotations on binary trees [17, 25].

This leads to structural and algorithmic questions: (1) Can any structure of a certain class be transformed into any other using the given flip operation? (2) What is the worst-case number of flips required to perform such a transformation? (3) Given two specific structures, how can we compute the minimum number of flips needed to reconfigure one into the other? These questions can also be rephrased in terms of the flip graph, where vertices represent the structures and edges correspond to flips: (1) Is the flip graph *connected*? (2) What is the *diameter* of the flip graph? (3) What is the computational *complexity* of finding shortest paths between vertices in the flip graph?

A central example is the reconfiguration of *matchings* in graphs, particularly *perfect matchings*, which are sets of pairwise non-adjacent edges covering every vertex. In this setting, a small local change corresponds to removing a set of edges and inserting another along an alternating cycle, while maintaining a valid matching throughout.

We study the reconfiguration of *odd matchings*, which are matchings that cover all but one vertex in a graph. This variant arises naturally in graphs without perfect matchings due to parity constraints, or in point sets of odd cardinality. In such cases, we refer to the graph or point set as being of odd *order*, where the order is the number of vertices or points. Each flip adds an edge between the unmatched vertex (which we also refer to as *isolated*) and another vertex v and removes the matching edge vw that was previously incident to v. This way, w becomes the new isolated vertex. We consider two distinct settings: In the *geometric setting*, the vertices are points in the plane, and matchings consist of straight lines between points that do not share endpoints and that do not cross. Each such *geometric flip* yields a (plane) odd matching. In the *combinatorial setting*, we consider odd matchings in an input graph G, and a *combinatorial flip* consists of removing and adding an edge of G such that the result is again an odd matching. For an illustration of both settings, see Figure 1.

Problem description

We study the problem Flipping Odd Matchings, which concerns reconfiguring one odd matching into another through a sequence of valid flip operations. We examine the problem in both the geometric and the combinatorial setting with their corresponding flip operations. The associated decision problem asks whether, given two odd matchings M and M' and an integer $k \in \mathbb{N}$, there exists a sequence of at most k flip operations that transforms M into M' within the respective setting.

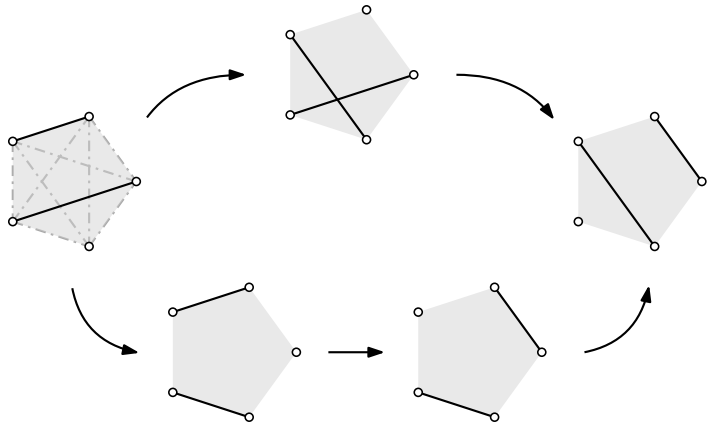


Figure 1 This figure illustrates the two distinct settings and their corresponding flip types. The odd matching on the left must be reconfigured into the odd matching on the right. The clockwise flip sequence represents a combinatorial reconfiguration, while the counterclockwise sequence is geometric. Dotted-dashed lines in the initial matching indicate additional edges present in the graph.

Our contribution

In this paper, we present a number of novel results on reconfiguring odd matchings in geometric and combinatorial settings. In particular, we provide a complete characterization of graphs in which each two odd matchings can be flipped into each other. Moreover, if the corresponding (combinatorial) flip graph is connected, its diameter is linear in the number of vertices of the input graph. Furthermore, we show that the decision variant of Flipping Odd Matchings is NP-complete in both considered settings, but fixed-parameter tractable in the case of convex point sets when parameterized in the flip distance.

A detailed summary of our results is given in Table 1.

Table 1 Current status of reconfiguring odd matchings; new results are highlighted in light blue.

	Setting	Connectedness	Diameter	Complexity
Geometric	Convex	Yes [2]	at most $3n/2 - \mathcal{O}(1)$, Corollary 16 at least $n - 2$ [2]	FPT, Theorem 13
	General position	Yes [2]	$\Omega(n)$ and $\mathcal{O}(n^2)$ [2]	NP-complete, Theorem 18
Combinatorial	Rectangular grids	Yes, Remark 5	$\Theta(n)$, Theorem 10	NP-complete, Theorem 23
	Planar graphs	Characterization, Theorem 3	if connected: $\Theta(n)$, Theorem 10	NP-complete, Theorem 23
	General graphs	Characterization, Theorem 3	if connected: $\Theta(n)$, Theorem 10	NP-complete, Theorem 23

Due to space constraints, full technical details and proofs for statements marked with ($\star$) are given in the full version of our paper [1].

Related work

In the geometric setting, it was recently shown in [2] that for a set of n points in general position in the plane, the flip graph of odd matchings is connected, with diameter bounded between $\Omega(n)$ and $\mathcal{O}(n^2)$. In contrast, for perfect matchings in the geometric setting, the connectedness of the flip graph remains an open question when flips are restricted to alternating cycles of sublinear length. However, if no such restriction is imposed, the flip graph is known to be connected [16]. Recent work shows that finding shortest flip sequences is NP-hard for point sets in general position, when the flip size is restricted to 4-cycles [6].

In the combinatorial setting, the connectivity of the flip graph for perfect matchings has been studied in [7]. The authors showed that the problem is PSPACE-complete for several graph classes, including split graphs and bipartite graphs of bounded bandwidth with maximum degree five. In contrast, they proved that it can be solved in polynomial time for strongly orderable graphs, outerplanar graphs, and cographs. In these positive cases, they also provided flip sequences of linear length.

Odd matchings have also been studied in the combinatorial setting, particularly in the special case of the sliding block puzzle known as *Gourds* [11, 19]. Here, the underlying graph is a triangular grid graph, where matching edges are assigned distinct colors, and in each flip, the added edge inherits the color of the removed edge. The connectedness of the flip graph is fully characterized when the underlying graph is a solid triangular grid graph [11]. For triangular grid graphs with holes, sufficient conditions for connectedness are known, but a complete characterization remains open [19].

We refer to [8] for a survey of additional results and open questions on the reconfiguration of planar graphs in both the combinatorial and geometric settings. Reconfiguration in general is also closely linked to *Gray codes* [22], which aim to list all feasible configurations in an order such that each configuration differs from the previous one only by a single flip. Such an ordering corresponds to identifying a Hamiltonian cycle in the flip graph, providing a compact and systematic traversal of the solution space [14, 15, 17, 24].

2 Preliminaries and basic observations

Geometric and combinatorial settings. We consider odd matchings in two different settings. In the *geometric setting*, let S be a set of n points embedded in the plane. A line segment between two points of S is an *edge*. A *matching* on S is a set of edges whose endpoints are pairwise distinct, said to be *plane* if no two edges cross. An *odd matching* M on S is a plane matching on S where exactly one point in S is not an endpoint of any edge of M. A *geometric flip* in an odd matching removes one edge and inserts another one so that the result remains an plane matching. In essence, an odd matching on a point set S is simply a plane odd matching on the complete graph with vertex set S.

In the *combinatorial setting*, we consider odd matchings on a fixed given graph G. A *matching* M in a simple graph $G = (V, E)$ is a set of pairwise disjoint edges, that is, no two edges in M share a common vertex. It is called *odd* if exactly one vertex of G is not incident to an edge in M. A *combinatorial flip* in an odd matching M removes one edge and replaces it with another from G, preserving the property of being an odd matching. In both settings, the new edge has to be incident to the isolated vertex of the previous odd matching.

The *flip graph* of a graph G (or point set S) has as its vertices the odd matchings of G (resp., S), with edges connecting odd matchings that differ in a single flip. We say that an odd matching M can be *reconfigured* to another odd matching M' if there exists a flip sequence transforming M into M', i.e., there is a path between them in the flip graph.

A graph or point set is said to be *reconfigurable* if its flip graph is connected, meaning any two odd matchings can be reconfigured into one another. The *flip distance* $d(M, M')$ between two matchings M and M' is the length of a shortest path between them in the flip graph. The *diameter* of the flip graph is the maximum flip distance over all pairs of odd matchings. Its *radius* is defined as $\min_M \max_{M'} d(M, M')$, and this minimum is attained by odd matchings that minimize their farthest distance to any other odd matching. Note that the diameter is at least the radius and at most twice the radius.

Union and symmetric difference of matchings in both settings. Let M and M' be odd matchings. An *M-alternating path* is a path that alternates edges in M and not in M; M-alternating cycles are defined analogously. Then $M \cup M'$ is a disjoint union of

- a path of even length (possibly zero) that connects the isolated vertex in M to the isolated vertex in M' and alternates edges of M' and M,
- cycles of even length alternating edges of M and M', and
- edges that lie in both M and M', which we call *happy edges*.

The *symmetric difference* $M \triangle M'$ is the set of edges that appear in only one of the two odd matchings. In other words, it is obtained from $M \cup M'$ by removing all happy edges.

Flipping along alternating paths. We consider the combinatorial setting. Let P be an M-alternating path from the isolated vertex v_M to some vertex v, where the last edge belongs to M. We can flip the matching along P, hence replacing $M \cap P$ by the edges in $P \setminus M$; the new isolated vertex is v. Next, we consider M-alternating paths to cycles in $M \triangle M'$.

▶ **Lemma 1.** *Let M and M' be odd matchings of a graph G. Suppose M can be reconfigured to M'. For every cycle C in $M \triangle M'$, there exists a vertex $v \in V(C)$ such that v has an M-alternating path to the isolated vertex v_M.*

Proof. Let C be a cycle in $M \triangle M'$. The edges in $M \cap C$ are eventually flipped to reconfigure M into M'. After flipping such an edge $e = vw$, one of its endpoints, say v, becomes isolated in the corresponding intermediate matching. Consequently, $G - v$ has a perfect matching. Hence, $M \setminus \{e\}$ is not maximal in $G - v$. By Berge's theorem [5], there exists an M-alternating path from w to v_M. ◀

Consider a cycle C in $M \triangle M'$ and assume that there exists an M-alternating path P from v_M to $v \in C$. Then $M \cap C$ can be replaced by $M' \cap C$ by flipping the edges along P, around C, and then restoring the edges on P. After this, v_M is again the isolated vertex. When applying the described flipping procedure in the special case that v_M is adjacent to a vertex $v \in C$, we say that we *switch* C. If C has k edges, then it takes $k+1$ flips to switch C. The additional flip occurs because we need to first place the isolated vertex on the cycle. This also holds for the geometric setting if C is planar; see Figure 2 for an illustration.

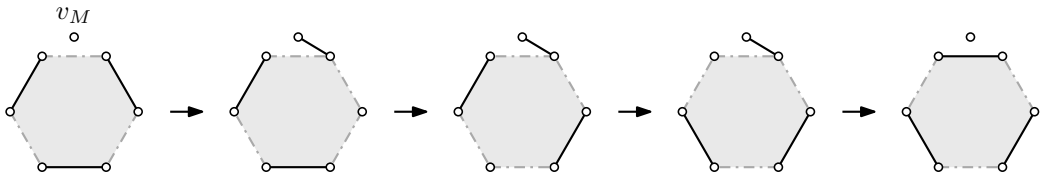


Figure 2 How to switch an alternating cycle.

However, this does not hold in general in the geometric setting. Suppose we wish to switch the cycle illustrated in Figure 3. The only feasible flips initially remove the blue edge "closest" to v_M. However, regardless of how this edge is removed, a red edge cannot be added immediately in the next flip, as doing so would cause it to cross a remaining blue edge.

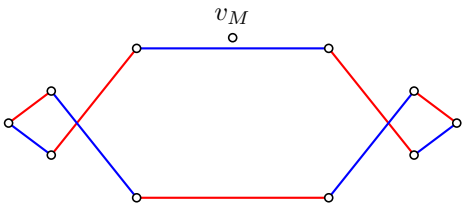


Figure 3 A non-plane alternating cycle in the geometric setting.

3 Reachability

We consider the question of whether two odd matchings of a given graph can be reconfigured into each other. For the geometric setting this has been proven to be always possible [2]. In contrast, the situation in the combinatorial setting is fundamentally different, as there exist simple examples where reconfiguration is not possible, as illustrated in Figure 4.

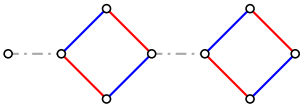


Figure 4 A connected graph with two odd matchings (red and blue, respectively) that cannot be reconfigured into each another. Specifically, the rightmost cycle cannot be reconfigured. This example can be extended for any odd n by increasing the size of the rightmost cycle.

In this section, we characterize the graphs that are reconfigurable, that is, for which the flip graph is connected. Further, we give a simple polynomial-time algorithm to decide whether a graph is reconfigurable.

▶ **Lemma 2.** *Let G be a graph of odd order, and consider an edge e.*
1. *Suppose that e is not contained in any odd matching of G. Then G is reconfigurable if and only if $G - e$ is reconfigurable.*
2. *Suppose that $e = uv$ is contained in every odd matching of G. Then G is reconfigurable if and only if $G - u - v$ is reconfigurable.*

Proof. If e is not contained in any odd matching of G, then clearly G is reconfigurable if and only if $G - e$ is, as e never appears in a flip sequence in G. Now suppose that $e = uv$ is contained in every odd matching of G. As no reconfiguration sequence in G ever flips e, it is easy to see that G is reconfigurable if and only if $G - u - v$ is reconfigurable. ◀

▶ **Theorem 3.** *Let G be a graph of odd order. Then G is reconfigurable if and only if for every edge $e = uv \in E(G)$, one of the following holds:*
1. *the edge e is either contained in all or in no odd matchings of G, or*
2. *at least one of $G - u$ and $G - v$ contains a perfect matching.*

Proof. By Lemma 2, we may assume that every edge of G is contained in some, but not all odd matchings of G. First assume that for every edge $e = uv \in E(G)$, $G - u$ or $G - v$ contains a perfect matching. Let M and M' be odd matchings in G with isolated vertices

v_M and $v_{M'}$, respectively. Flipping edges along the path in $M \triangle M'$ maps v_M to $v_{M'}$, so assume $v_M = v_{M'}$. Then $M \triangle M'$ is a collection of cycles. Let C be a cycle in $M \triangle M'$ and let $e = uv \in M$ be an edge on C. By assumption, $G - u$ or $G - v$ contains a perfect matching, say $G - u$. By applying Berge's theorem [5] to $M \setminus \{e\}$ in $G - u$, there exists an M-alternating path from v_M to v. Using this alternating path to switch C yields a matching M_1 with $|M_1 \triangle M'| < |M \triangle M'|$. Inductively, this yields a flip sequence from M to M'.

Now assume that G contains an edge $e = uv$ such that $G - u$ and $G - v$ do not have a perfect matching. In other words, u and v are covered in all odd matchings. By assumption, there exists an odd matching M_1 containing e and an odd matching M_2 not containing e. To reconfigure M_1 to M_2, we eventually need to perform a flip involving e. However, this isolates u or v, which is a contradiction. Hence, G is not reconfigurable. ◀

▶ Remark 4. Let G be a factor-critical graph, i.e., $G - v$ has a perfect matching for every $v \in V(G)$. Clearly, G then satisfies condition (2) in Theorem 3, and hence G is reconfigurable.

▶ Remark 5 (⋆). For any odd values $w, h \in \mathbb{N}$ the $w \times h$ square grid graph, i.e., the Cartesian product $P_w \square P_h$, is reconfigurable.

There is a straightforward polynomial-time algorithm to check the conditions of the characterization (Theorem 3), i.e., deciding whether a graph is reconfigurable.

▶ **Theorem 6** (⋆). *Given a graph G, we can determine whether the flip graph of odd matchings in G is connected in $\mathcal{O}(n^{4.5})$ time. In other words, the reconfigurability of a graph can be checked in polynomial time.*

4 Radius and diameter of the flip graph

We now study the diameter of the flip graphs in both combinatorial and geometric settings.

4.1 Linear diameter of the flip graph in general graphs

In this section, we show that the diameter of the flip graph in the combinatorial setting is linear in the order n of the input graph. Given an input odd matching M and a target matching M' in a graph G of order n, such that it is possible to reconfigure M to M', the idea is to switch the cycles in $M \triangle M'$, traversing them in a tree-like fashion. In order to do so, we use an auxiliary tree structure. The proof proceeds in two steps. In the first part, we construct an auxiliary graph for the cycle traversal. In the second part, we prove that there exists a flip sequence of length linear in n reconfiguring M into M'. In the following, we always assume that the isolated vertices of M and M' coincide. This will later be achieved by first flipping the path in $M \triangle M'$ between these two vertices.

Construction of the auxiliary graph

We construct an auxiliary directed graph $\overrightarrow{H}$ that will be used to construct the flip sequence. At any step of the construction, let H denote the underlying undirected graph of $\overrightarrow{H}$.

We initialize $\overrightarrow{H}$ by introducing, for each vertex v of G, two corresponding vertices in $\overrightarrow{H}$: an *in-vertex* v_{in} and an *out-vertex* v_{out}; for an illustration see Figure 5. For every edge uv in M', we add the arcs $(u_{\text{in}}, v_{\text{out}})$ and $(v_{\text{in}}, u_{\text{out}})$ to $\overrightarrow{H}$. Note that $v_{M,\text{out}}$ remains isolated throughout the process, so we refer to the in-vertex $v_{M,\text{in}}$ simply as v_M.

Let t be the number of cycles in $M \triangle M'$. In the following, we modify $\overrightarrow{H}$ by adding arcs. To this end, we iteratively construct auxiliary odd matchings $M_0, \ldots, M_t$ in G. Set $M_0 := M$. Suppose M_i, for some $i \in \{0, \ldots, t-1\}$, is the last constructed matching. We now *process* M_i

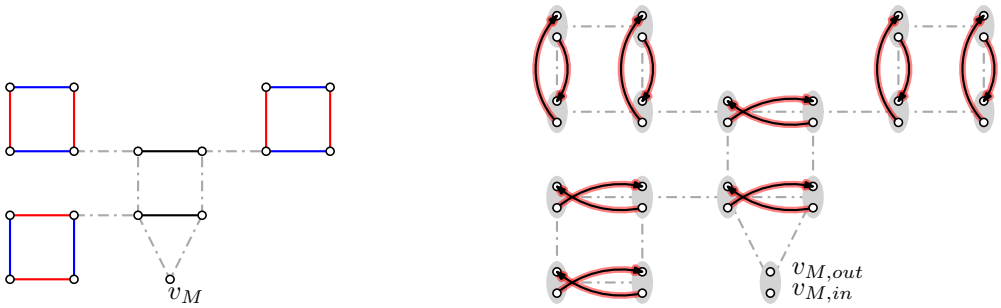


Figure 5 (a) A graph G with the initial and target matching M and M', respectively, and (b) the initialization of the auxiliary graph $\overrightarrow{H}$.

to construct M_{i+1}. To this end, we choose an arbitrary cycle K in $M_i \triangle M'$. By Lemma 1, G contains an M_i-alternating path P from v_M to K. Let u_{C_i} be the first vertex on a cycle C_i in $M_i \triangle M'$ that we encounter when traversing P starting from v_M. We now traverse P from u_{C_i} towards v_M. Suppose that $e = ww'$ is the current edge on the path and assume that w' is closer to v_M on P than w. If $e \in M_i$, we simply move on to the next edge in P. If $e \notin M_i$, we add the arc $(w_{\text{out}}, w'_{\text{in}})$ to $\overrightarrow{H}$. If there is no directed path from w'_{in} to v_M in $\overrightarrow{H}$, we traverse the next edge on P. Otherwise, we stop the traversal and define M_{i+1} as the matching obtained from M_i by replacing $M_i \cap C_i$ by $M' \cap C_i$. One such iteration is visualized in Figure 6.

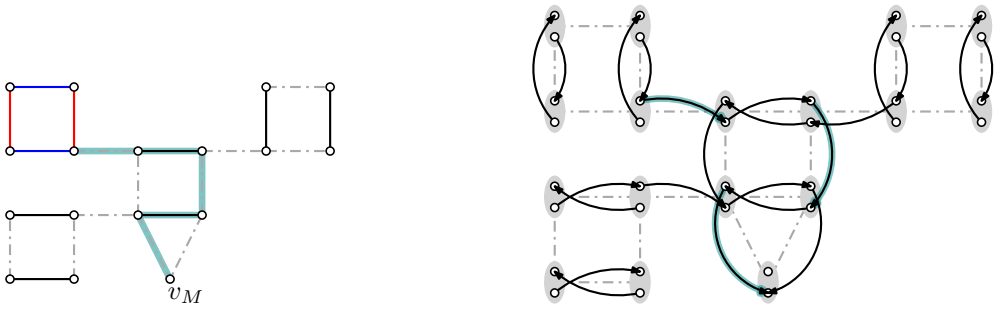


Figure 6 Iteration of the construction of $\overrightarrow{H}$, at the moment of inclusion of the top-left cycle. (a) The auxiliary matching M_i with the highlighted alternating path connecting the cycle to v_M. (b) The corresponding state of $\overrightarrow{H}$ and the highlighted corresponding arcs added to $\overrightarrow{H}$.

We now collect some observations on the structure of $\overrightarrow{H}$ and the matchings $M_0, \ldots, M_t$.

▶ **Lemma 7** (⋆). *We have $M_t = M'$ and for all $i \in \{0, \ldots, t\}$, v_M is isolated in M_i.*
The arcs of $\overrightarrow{H}$ always join an in-vertex and an out-vertex. In particular, H is bipartite.
Each out-vertex in $\overrightarrow{H}$ has at most one outgoing arc.

In particular, Lemma 7 implies the following:

▶ **Corollary 8.** *The graph H is a forest. Further, let $\mathcal{C}$ be the connected component of H containing v_M. In $\overrightarrow{H}$, there is a directed path from every vertex in $\mathcal{C}$ to v_M. Moreover, for every cycle K in $M \triangle M'$, $\mathcal{C}$ contains the vertex $u_{K,out}$.*

▶ **Lemma 9** (⋆). *If (u_{in}, v_{out}) is an arc in $\overrightarrow{H}$, then $uv \in M'$. If (u_{out}, v_{in}) is an arc in $\overrightarrow{H}$, then $uv \in E(G) \setminus M'$. Hence, every directed path in $\overrightarrow{H}$ defines an M'-alternating path in G.*

Description of the flipping procedure

Using the graph H above, we now describe a flip sequence that transforms M to M'.

Let $\mathcal{C}$ be the connected component of H containing $v_M = v_{M,\text{in}}$. We traverse $\mathcal{C}$ in a depth-first search fashion, starting and ending at v_M. In parallel, we perform certain flip operations in G. We maintain the following property: whenever a vertex v_{in} is visited in the traversal, the corresponding vertex v is isolated in G.

Let v_{in} be the currently visited vertex. Let $v_{1,\text{out}}, \dots, v_{k,\text{out}}$ be all vertices of $\overrightarrow{H}$ such that $(v_{i,\text{out}}, v_{\text{in}})$ is an arc. Two cases can occur:

(1) All vertices $v_{1,\text{out}}, \dots, v_{k,\text{out}}$ have already been visited. Then we backtrack. That is, if $v_{\text{in}} = v_M$, we terminate. Otherwise, let $w_{1,\text{out}}, w_{2,\text{in}}$ be the next vertices on the directed path from v_{in} to v_M. We will show that the edge w_1w_2 is present in G. We flip it to vw_1, so that the isolated vertex becomes w_2, and next visit $w_{2,\text{in}}$.

(2) There is an unvisited vertex $v_{i,\text{out}}$. In this case, we check whether v_i is contained in a cycle C in $M \triangle M'$ that has not yet been switched, and if so, switch C. Subsequently, we consider the second neighbor v'_{in} of $v_{i,\text{out}}$ in H. We will show that the edge $v'v_i$ is present in G. We flip it to v_iv, so that the isolated vertex becomes v'. We mark $v_{i,\text{out}}$ as visited and then visit v'_{in}.

▶ **Theorem 10** (⋆). *Let M and M' be odd matchings of a graph G such that M can be reconfigured to M'. Using the above procedure, we can reconfigure M to M' using $\mathcal{O}(n)$ flips.*

4.2 Convex point sets

We start with a refinement of the notion of happy edges in the case of convex point sets. *Good happy edges* are defined by the following recursive definition:

(1) Happy edges on the convex hull are necessarily good happy edges.

(2) If a happy edge e splits the convex point set into two parts such that one part contains the isolated vertex (possibly along with some edges) and all edges in the other part are happy edges (implying that they all are good happy edges), then e is a good happy edge.

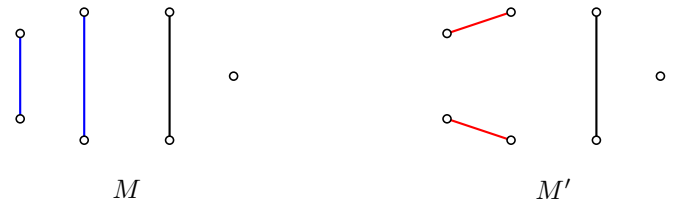


Figure 7 Example of a convex point set in which a happy edge needs to be flipped.

A happy edge that is not good is a *bad* happy edge. Figure 7 provides an example in which a happy edge needs to be flipped. However, this edge is not a good happy edge. On the contrary, good happy edges are preserved in shortest flip sequences.

▶ **Lemma 11** (⋆). *No shortest flip sequence between any two odd matchings on a convex point set flips good happy edges.*

Our recursive definition of good happy edges allows us to construct an equivalent instance without happy edges in linear time.

▶ Remark 12 (⋆). Let M and M' be two matchings. We can remove good happy edges and their incident vertices in time that is linear in the number of vertices.

▶ **Theorem 13.** *The flip distance k between two odd matchings of the same convex point set S is fixed-parameter tractable in k.*

Proof. We remove all good happy edges using Remark 12. As all remaining edges need to be flipped at least once, the instance is a "no" instance, if there are more than k edges left. In each step, there are at most $2k$ endpoints of matching edges. Adding an edge between the isolated vertex and an endpoint determines which edge gets removed. Thus, we need to check at most $2k$ possible flips per step and at most $(2k)^k$ sequences. ◀

▶ Remark 14. Actually, Theorem 13 shows a slightly stronger property: The problem of determining whether the flip distance between two odd matchings is at most k admits a polynomial size kernel with at most k edges and at most $2k + 1$ vertices. That is, for every instance consisting of two matchings M and M' along with a parameter k, there exists a corresponding instance whose size is polynomial in k, such that this new instance is a `YES` instance if and only if the original instance is a `YES` instance.

Consider the union of an initial matching M and a target matching M'. With respect to this, let A be the set of all good happy edges, B the set of all bad happy edges, C the set of edges in M that lie on even alternating cycles, c the number of such cycles, and D the set of edges in M that lie on the alternating path. We have observed that all edges in C and D are flipped at least once, and all edges in B are flipped at least twice. Additionally, each even alternating cycle requires one extra flip to place the isolated vertex onto the cycle. The final flip in a cycle returns the isolated vertex to the component it originally came from. This yields a lower bound on the flip distance, which is tight when the union of M and M' is crossing-free.

▶ **Theorem 15 (⋆).** *Let M, M', A, B, C, c, and D be as defined above, and let $M \cup M'$ be crossing-free. Then the flip distance from M to M' is $2|B| + |C| + c + |D|$.*

We only give an intuition for the proof of Theorem 15. The idea is to place the isolated vertex once on every alternating cycle in $M \cup M'$ without making any unnecessary flips. In this step, it helps that in a convex point set if a vertex sees a connected component, it always sees two consecutive vertices. Therefore, we do not have to worry about parity constraints. Then, since the two matchings are together crossing-free, we can greedily switch every cycle in the optimal number of flips.

▶ **Corollary 16.** *The radius of the flip graph of odd matchings for $n = 2m + 1$ points in convex point sets is at most $\frac{3m}{2} - \mathcal{O}(1)$. As a consequence, the diameter of the flip graph is at most $3m - \mathcal{O}(1) = \frac{3n}{2} - \mathcal{O}(1)$.*

Proof. For any given matching M we show that we can flip M into a given matching M' that has all its edges on the convex hull. Observe that all happy edges of $M \cup M'$ are good happy edges, and that it contains at most $\frac{m}{2}$ alternating cycles and one alternating path. Plugging the parameters into Theorem 15 we obtain the desired upper bound. Since the diameter of a graph is bounded by twice its radius, we obtain an upper bound on the diameter of $3m - \mathcal{O}(1) = \frac{3n}{2} - \mathcal{O}(1)$. ◀

▶ Remark 17 (⋆). For any given odd matching M there exists a matching M' on $n = 2m + 1$ points in a convex point set such that $d(M, M') = \frac{3m}{2} - \mathcal{O}(1)$.

With this result, we improve the previously known upper bound on the diameter from $4m - \mathcal{O}(1) = 2n - \mathcal{O}(1)$, as given in [2], to $3m - \mathcal{O}(1) = \frac{3n}{2} - \mathcal{O}(1)$. By also establishing a matching lower bound on the radius, we demonstrate that this upper bound is best possible among all approaches that proceed via a canonical intermediate structure. Any further improvement to the upper bound on the diameter will likely require more sophisticated techniques, potentially exploiting structural similarities between M and M'.

5 Hardness results

In this section, we analyze the computational complexity of the problem in various settings, specifically for point sets in general position, grid graphs, and more broadly, planar graphs. We prove that deciding whether there exists a flip sequence of length at most k between two odd matchings is NP-complete in each of these settings.

5.1 NP-completeness in point sets

▶ **Theorem 18.** *Let M and M' be two odd matchings on a set of n points in the plane. Deciding whether there is a flip sequence of length k transforming M into M' is* NP*-complete.*

We reduce from the NP-complete problem Planar Monotone 3SAT [9]. Membership in NP follows easily from the fact that a valid flip sequence of length at most k serves as a certificate. As the flip graph of odd matchings in this setting has diameter $\mathcal{O}(n^2)$ as shown in [2], a valid flip sequence has polynomial length.

Planar Monotone 3SAT. This is a variant of 3SAT where each clause has at most three literals, either all positive or all negative. The variable-clause incidence graph is planar and must be embedded with variables on a horizontal line, positive clauses above, and negative clauses below. Moreover, every such instance can be represented in a rectilinear form [20], where each variable and clause corresponds to a rectangle, and their relationship is depicted by a vertical line segment connecting them; we refer to Figure 8 for an illustration.

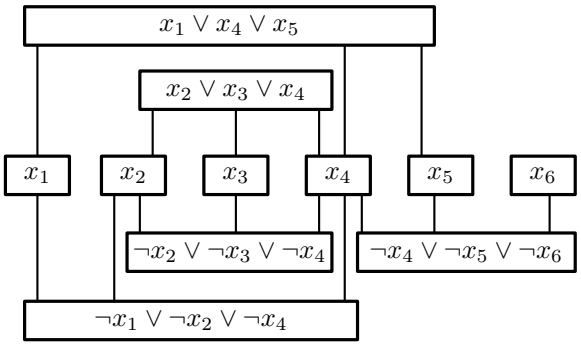


Figure 8 The clause-variable incidence graph of the Boolean formula Φ in conjunctive normal form, $\Phi = (x_1 \vee x_4 \vee x_5) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_4 \vee \neg x_5 \vee \neg x_6)$ as an instance of Planar Monotone 3SAT.

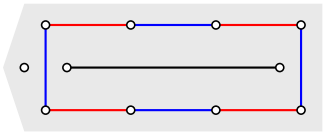


Figure 9 The variable gadget with the isolated vertex on the left.

Variable gadget. The variable gadget is shown in Figure 9. While the black edge is happy, the blue and red edges are only contained in the initial and target configuration, respectively.

There are two optimal flip sequences that transform the variable gadget from its initial state to its target state. Two representative intermediate configurations are shown in Figure 10. The direction of traversal encodes the truth value assigned to the variable.

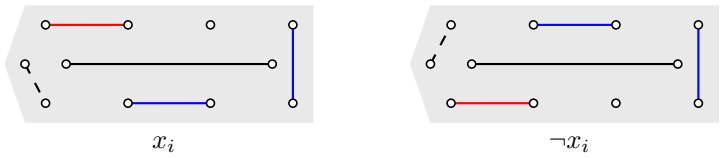


Figure 10 The variable gadget with a truth assignment: true on the left, false on the right.

Clause gadget. The clause gadget is depicted in Figure 11; note that the figure also includes three variable gadgets to illustrate the overall embedding. As before, blue and red edges indicate the initial and target state, respectively. If a clause contains only positive literals, we place it above the corresponding variable gadgets, and if it contains only negative literals, we place it below. Additionally, we introduce edges, shown in black, which are part of both matchings; they serve to obstruct visibility between the vertices of the variable and clause gadgets, permitting interaction only through a single designated pair.

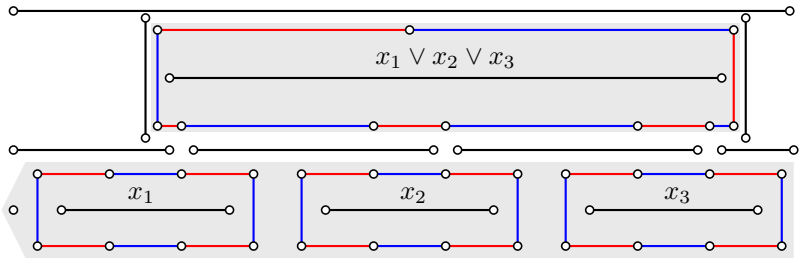


Figure 11 Clause gadget corresponding to $(x_1 \vee x_2 \vee x_3)$, with the associated variable gadgets included for clarity.

Figure 12 illustrates how a valid truth assignment enables interaction with a clause gadget. During an optimal reconfiguration of a variable gadget, there is a point in the sequence where an isolated vertex becomes visible to a vertex in the clause gadget if and only if the chosen sequence reflects a valid truth assignment. Moreover, when the clause gadget is reconfigured optimally, no isolated vertex within the clause gadget will ever have visibility to a vertex in any other variable gadget.

Rectilinear representation. We complete the construction using the rectilinear representation of the Planar Monotone 3SAT instance. Each variable and clause rectangle is replaced by the corresponding gadget. Happy edges are added to block unintended visibility, while vertical segments in the rectilinear layout define visibility gaps between designated vertices of variable and clause gadgets. We refer to Figures 8 and 13 for illustrations.

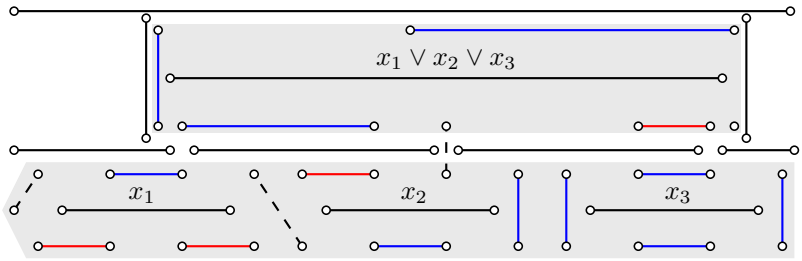


Figure 12 Clause gadget in an intermediate state. The clause $(x_1 \vee x_2 \vee x_3)$ is satisfied because the variable x_2 is assigned `TRUE`. In contrast, x_1 is assigned `FALSE` and therefore does not contribute to reconfiguring the clause.

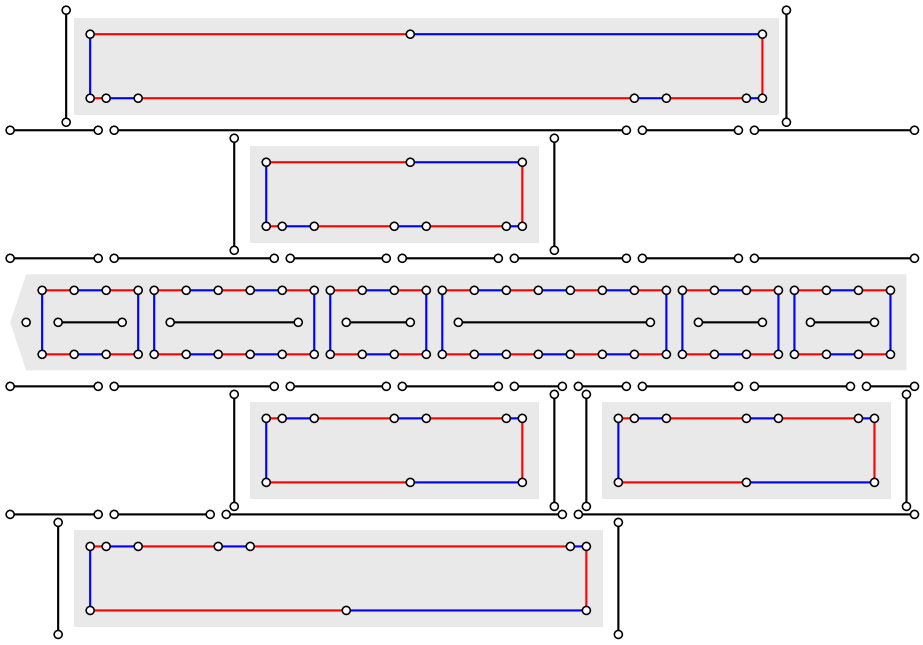


Figure 13 An instance of the odd matching reconfiguration problem, derived from the Planar Monotone 3SAT instance depicted in Figure 8.

▶ **Proposition 19** ($\star$). *Let Φ be an instance of Planar Monotone 3SAT with C clauses and V variables. There exists a flip sequence from M to M' (the corresponding odd matching instances) of length $\frac{1}{2}|M \triangle M'| + V + C$ flips if and only if Φ has a valid truth assignment.*

Theorem 18 follows from Proposition 19. In our reduction, we construct point sets that are not in general position, as illustrated in the figures. To address this, Lemma 20 shows how to transform the point set into general position while preserving the essential structure.

5.2 Generalizing to point sets in general position

This section is dedicated to show the following lemma.

▶ **Lemma 20.** *Let S be a point set that can be embedded in an $h \times w$ grid. Then there exists a mapping ϕ that maps each point of S to a point on an $f(h,w) \times g(h,w)$ grid where f and g are polynomial functions in h and w such that*

1. *for every three points p, q, r of S, $\phi(p), \phi(q)$, and $\phi(r)$ are not collinear,*
2. *if p, q, and r are three non-collinear points of S, then the orientation of the triplet (p, q, r) is the same as that of $\big(\phi(p), \phi(q), \phi(r)\big)$, and*
3. *for every two points p and q of S, $\phi(p)$ and $\phi(q)$ have different x- and y-coordinates.*

An ε-square is a square with side length ε. Our main idea is to place small ε-squares $B_\varepsilon(v)$ centered at each vertex v. These ε-squares are small enough such that for any triple of non-collinear points $p, q, r \in S$, a line $p'q'$ with $p' \in B_\varepsilon(p)$ and $q' \in B_\varepsilon(q)$ does not cross $B_\varepsilon(r)$. Further, we blow up our grid such that there are enough points inside each ε-square, and hence, there is at least one grid point that shares no x- or y- coordinate with any other point and that is not on a line spanned by two points of S.

▶ **Lemma 21** (⋆). *Let ℓ be a line through two points of an $h \times w$ square grid G. For every point p of G not on ℓ, let p^x and p^y be the point on ℓ that shares the same x- and y-coordinate with p, respectively. Then the distance from p to each of p^x and p^y is at least $\frac{1}{2hw}$.*

The lemma above is used to show the following lemma.

▶ **Lemma 22** (⋆). *Let p, q be two points of an $h \times w$ grid G. Let p' and q' be points in the ε-squares centered at p and q, respectively. If $\varepsilon \leq \frac{1}{16w^3h^3}$, then for every grid point r that does not lie on the line through p and q, the orientation of the triplet (p, q, r) is the same as (p', q', r') for every point r' in the ε-square centered at r.*

We are now ready to prove Lemma 20.

Proof of Lemma 20. Let S be a set of n points, and denote them by $p_1, \ldots, p_n$. Let $\varepsilon = \frac{1}{16h^3w^3}$. We construct the new grid G' such that for every vertex p of the point set, the ε-square $B_\varepsilon(p)$ centered at p contains $(n^4+n) \times (n^4+n)$ grid points. The resulting grid G' has now size $h \cdot 16h^3w^3 \cdot (n^4+n) \times w \cdot 16h^3w^3 \cdot (n^4+n)$ which is polynomial in h and w.

We inductively define the mapping ϕ. We maintain that for $i = 1, \ldots, n$, the set $S_i := \{\phi(p_1), \ldots, \phi(p_i)\}$ satisfies the conditions (1), (2), and (3) of the lemma.

We define $\phi(p_1) = p_1$. The set S_1 trivially satisfies (1)–(3). Suppose we have constructed the set S_i for some $i \in \{1, \ldots, n-1\}$. We now construct the set S_{i+1}. Consider $\binom{i}{2}$ lines spanned by any two points in S_i and the i vertical lines and i horizontal lines passing through a point in S_i. The former $\binom{i}{2}$ lines intersect in less than $(i^2)^2 < n^4$ grid points in $B_\varepsilon(p_{i+1})$. Hence, even when we avoid the i vertical and i horizontal lines, we can choose one point in $B_\varepsilon(p_{i+1})$ to be $\phi(p_{i+1})$ such that S_{i+1} satisfies (1) and (3). It remains to prove that S_{i+1} also satisfies (2). It is sufficient to consider a triplet (p_a, p_b, p_{i+1}) for $1 \leq a < b \leq i$. By construction, $\phi(p_a) \in B_\varepsilon(p_a)$ and $\phi(p_b) \in B_\varepsilon(p_b)$. Hence, by Lemma 22, the orientation of (p_a, p_b, p_{i+1}) is the same as that of $\big(\phi(p_a), \phi(p_b), \phi(p_{i+1})\big)$. The lemma then follows. ◀

5.3 NP-completeness in square grid graphs

A *rectilinear Steiner tree* is a tree that connects a given set K of points in the plane by only using horizontal and vertical line segments. The Rectilinear Steiner Tree Problem [10] asks whether there exists a tree that connects a given set of vertices using only horizontal and vertical line segments, such that the total edge length does not exceed a specified bound.

The problem is known to be strongly NP-hard; that is, it remains NP-hard even when all vertices lie on a grid of polynomial size. Moreover, there always exists an optimal rectilinear Steiner tree contained within the Hanan grid [12], which is constructed by extending horizontal and vertical lines from each vertex in the input set. This result also implies that the grid can be arbitrarily refined if needed, without affecting the edges of an optimal solution.

We reduce from the Rectilinear Steiner Tree Problem to obtain the following.

▶ **Theorem 23.** *Deciding whether there exists a flip sequence of length ℓ between two odd matchings in grid graphs is* NP*-hard (and consequently in planar and general graphs as well).*

High-level overview. Let K be a set of k points in an $n \times n$ grid. We assume, without loss of generality, that $n > k$. We aim to construct two odd matchings on the grid graph such that any short flip sequence between them corresponds to a rectilinear Steiner tree of short total edge length. Our reduction works in two phases: (1) By leveraging the structure of the Hanan grid, we restrict the input to point sets whose coordinates are all congruent to 1 mod $8n$, and (2) by carefully constructing two odd matchings on a sufficiently large grid graph, we ensure that short flip sequences between them correspond to short trees.

Construction. Let v_{low} be a point in K with the smallest y-coordinate (break ties arbitrarily). Without loss of generality, assume that its y-coordinate is 0. We embed the point set K into an $8n \cdot (n + 4n^2) \times 8n \cdot (n + 4n^2)$ grid graph G'; in particular, each point $v_i = (x_i, y_i) \in K$ is mapped to the vertex $(8n \cdot x_i + 1,\ 8n \cdot y_i + 1)$. We obtain a new set of points denoted by K'.

▶ **Lemma 24** ($\star$). *K can be connected by an RST of length ℓ if and only if K' can be connected by an RST of length $8n\ell$.*

Next, we replace every vertex in G' as follows: if a vertex belongs to K, we replace it by eight vertices arranged in an alternating cycle of four edges from M and M'; otherwise, we replace it with eight vertices and four happy edges; see Figure 14 for an illustration.



Figure 14 Vertex gadgets for vertices in K (left), and not in K (right).

Then we add an additional row and an additional column to the bottom and to the right. We place the isolated vertex of both M and M' below the third column of the gadget that corresponds to v_{low}. For the remaining points there is a unique way to connect them to an odd matching. The construction is illustrated in Figure 15. We remark that in order to keep the instance smaller, we do not use the refined version of the grid (i.e., G') in the illustration, we also do not add the additional $4n^2$ vertices to the right and top of the relevant area.

▶ **Proposition 25** ($\star$). *For any instance of RSTP with k vertices on an $n \times n$ grid, there exists an RST of total edge length at most ℓ if and only if there exists a flip sequence for two corresponding odd matchings M, M' transforming M into M' of length at most $4 \cdot 8n \cdot \ell + 8k$.*

With this, Theorem 23 can be easily obtained from Proposition 25.

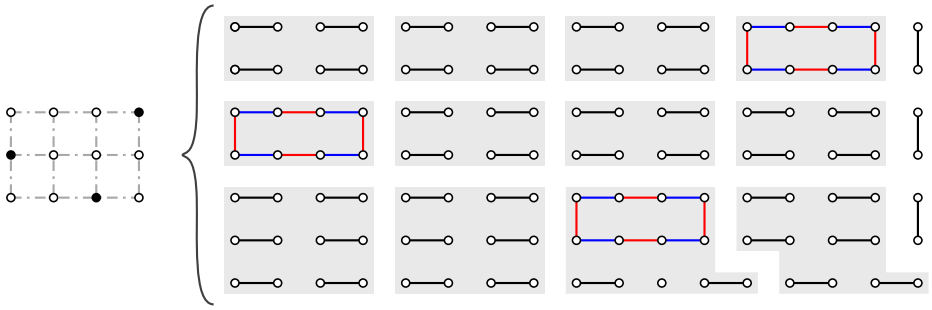


Figure 15 Reduction from rectilinear Steiner tree, massively scaled down for illustration.

6 Conclusions and future work

In this work, we investigated the reconfiguration of odd matchings and provided a complete characterization of when the associated flip graph is connected. This complements a recent result that showed connectedness of the flip graph in the geometric setting [2]. We further showed that, in the combinatorial setting, each connected component of the flip graph has diameter linear in the cardinality of the matchings. We established that the problem is NP-complete, both in general graphs (in particular, even in rectangular grid graphs) and in geometric settings. For the geometric setting, we developed a framework to embed partial order types into general position. We believe that our framework can be applied in several similar proofs. Lastly, for point sets in convex position, we provided a fixed-parameter tractable algorithm and improved the upper bound on the diameter of the flip graph.

The following questions are related to our research.

- Considering that planarity constraints make things harder: Can the linear bound for the diameter of the flip graph in the combinatorial setting be generalized to the geometric setting? Or can it be used to derive bounds on the diameter of the flip graph in the geometric setting?
- What is the complexity of finding shortest flip sequences between odd matchings in convex point sets? Closing the gap between the upper bound and the lower bound of the diameter can be seen as a first promising direction.
- Now that we established hardness of the flip distance problem: What can be said about fixed-parameter tractability and approximability of flip distances in both the combinatorial and the general geometric setting?

References

1 Oswin Aichholzer, Sofia Brenner, Joseph Dorfer, Hung P. Hoang, Daniel Perz, Christian Rieck, and Francesco Verciani. Flipping odd matchings in geometric and combinatorial settings, 2025. arXiv:2508.18457.

2 Oswin Aichholzer, Anna Brötzner, Daniel Perz, and Patrick Schnider. Flips in odd matchings. *Computational Geometry*, 129:102184, 2025. doi:10.1016/j.comgeo.2025.102184.

3 Sepehr Assadi, Soheil Behnezhad, Sayan Bhattacharya, Martín Costa, Shay Solomon, and Tianyi Zhang. Vizing's theorem in near-linear time. In *Symposium on Theory of Computing (STOC)*, pages 24–35, 2025. doi:10.1145/3717823.3718265.

4 David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21–46, 1996. doi:10.1016/0166-218X(95)00026-N.

5 Claude Berge. Two theorems in graph theory. *Proceedings of the National Academy of Sciences of the United States of America*, 43:842–844, 1957. doi:10.1073/pnas.43.9.842.

6 Carla Binucci, Fabrizio Montecchiani, Daniel Perz, and Alessandra Tappini. Flipping matchings is hard, 2025. doi:10.48550/arXiv.2503.02842.

7 Marthe Bonamy, Nicolas Bousquet, Marc Heinrich, Takehiro Ito, Yusuke Kobayashi, Arnaud Mary, Moritz Mühlenthaler, and Kunihiro Wasa. The perfect matching reconfiguration problem. In *International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 80:1–80:14, 2019. doi:10.4230/LIPICS.MFCS.2019.80.

8 Prosenjit Bose and Ferran Hurtado. Flips in planar graphs. *Computational Geometry*, 42(1):60–80, 2009. doi:10.1016/j.comgeo.2008.04.001.

9 Mark de Berg and Amirali Khosravi. Optimal binary space partitions for segments in the plane. *International Journal of Computational Geometry & Applications*, 22(3):187–206, 2012. doi:10.1142/S0218195912500045.

10 M. R. Garey and David S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977. doi:10.1137/0132071.

11 Joep Hamersma, Marc J. van Kreveld, Yushi Uno, and Tom C. van der Zanden. Gourds: A sliding-block puzzle with turning. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 33:1–33:16, 2020. doi:10.4230/LIPICS.ISAAC.2020.33.

12 Maurice Hanan. On Steiner's problem with rectilinear distance. *SIAM Journal on Applied Mathematics*, 14(2):255–265, 1966. doi:10.1137/0114025.

13 Percy J. Heawood. Map-colour theorem. *Proceedings of the London Mathematical Society*, 51:161–175, 1949. doi:10.1112/plms/s2-51.3.161.

14 M. Carmen Hernando, Ferran Hurtado, Alberto Márquez, Mercè Mora, and Marc Noy. Geometric tree graphs of points in convex position. *Discrete Applied Mathematics*, 93(1):51–66, 1999. doi:10.1016/S0166-218X(99)00006-2.

15 M. Carmen Hernando, Ferran Hurtado, and Marc Noy. Graphs of non-crossing perfect matchings. *Graphs and Combinatorics*, 18:517–532, 2002. doi:10.1007/s003730200038.

16 Michael E. Houle, Ferran Hurtado, Marc Noy, and Eduardo Rivera-Campo. Graphs of triangulations and perfect matchings. *Graphs and Combinatorics*, 21:325–331, 2005. doi:10.1007/s00373-005-0615-2.

17 Ferran Hurtado and Marc Noy. Graph of triangulations of a convex polygon and tree of triangulations. *Computational Geometry*, 13(3):179–188, 1999. doi:10.1016/S0925-7721(99)00016-4.

18 Takehiro Ito, Erik D. Demaine, Nicholas J. A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12-14):1054–1065, 2011. doi:10.1016/J.TCS.2010.12.005.

19 Naonori Kakimura and Yuta Mishima. Reconfiguration of labeled matchings in triangular grid graphs. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 43:1–43:16, 2024. doi:10.4230/LIPICS.ISAAC.2024.43.

20 Donald E. Knuth and Arvind Raghunathan. The problem of compatible representatives, 1992. doi:10.48550/arXiv.cs/9301116.

21 Shen Lin and Brian W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973. doi:10.1287/OPRE.21.2.498.

22 Torsten Mütze. Combinatorial gray codes – an updated survey. *Electronic Journal of Combinatorics*, 2024. Dynamic survey. doi:10.37236/11023.

23 Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018. doi:10.3390/A11040052.

24 Eduardo Rivera-Campo and Virginia Urrutia-Galicia. Hamilton cycles in the path graph of a set of points in convex position. *Computational Geometry*, 18(2):65–72, 2001. doi:10.1016/S0925-7721(00)00026-2.

25 Daniel D. Sleator, Robert E. Tarjan, and William P. Thurston. Rotation distance, triangulations, and hyperbolic geometry. *Journal of the American Mathematical Society*, 1(3):647–681, 1988. doi:10.2307/1990951.

26 Jan van den Heuvel. The complexity of change. In *Surveys in Combinatorics 2013*, volume 409 of *London Mathematical Society Lecture Note Series*, pages 127–160. 2013. doi:10.1017/CBO9781139506748.005.

27 Vadim G. Vizing. Critical graphs with given chromatic class (in russian). *Metody Discret. Analiz.*, 5:9–17, 1965.

Separability of Witness Gabriel Drawings

Carolina Haase
Trier University, Germany

Philipp Kindermann
Trier University, Germany

William Lenhart
Williams College, Williamstown, MA, USA

Giuseppe Liotta
University of Perugia, Italy

Abstract

A witness Gabriel drawing Γ is a straight-line drawing of a graph in which any two vertices of Γ are adjacent if and only if the disk having these vertices as antipodal points contains no element of a special set of points called witnesses. A witness Gabriel drawing is linearly separable if the vertices and the witnesses lie in opposite half-planes. We prove that every outerplanar graph has a linearly separable witness Gabriel drawing by introducing and studying a new type of drawing that we call a border parabola drawing. We then use border parabola drawings to characterize those triangle-free graphs that admit a linearly separable witness Gabriel drawing. We also consider witness Gabriel drawings where no witness lies in the interior of the convex hull of the vertex set, which we call convexly separable drawings. We construct witness Gabriel drawable graphs for which any witness Gabriel drawing must be convexly separable and that do not admit any linearly separable witness Gabriel drawing.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Mathematics of computing → Graph theory

Keywords and phrases Proximity Drawings, Witness Gabriel Graphs, Geometric Graph Theory

Digital Object Identifier 10.4230/LIPIcs.GD.2025.13

Funding Research of Giuseppe Liotta supported in part by MUR PON Proj. ARS01 00540 and by MUR PRIN Project no. 2022TS4Y3N.

1 Introduction

A proximity graph of a set of distinct points S in the Euclidean plane is a geometric graph G whose vertex set is S. Any two vertices u and v of G are adjacent if and only if no other vertex w of G is in the "neighborhood" of u and v. Depending on the definition of neighborhood, we have different types of proximity graphs. For example, in a *Gabriel graph* uv is an edge if and only if the *Gabriel disk*, i.e., the disk having u and v as antipodal points, does not contain any other vertex w. In a *rectangle of influence drawing* u and v are adjacent if and only if the *rectangle of influence* of u and v, i.e., the axis-parallel rectangle having u and v at opposite corners, does not contain any other vertex w.

For a given definition of proximity, different graph topologies correspond to distinct distributions of their vertex sets, making these graphs effective descriptors of the morphological properties of point configurations (see, e.g., [18,19]). As a consequence, proximity graphs find applications in a variety of contexts, including machine learning, wireless networks, computer vision, and business analytics (see, e.g., [5,16,19,21]). In information visualization, proximity graphs are used to measure the faithfulness of network layouts (see, e.g., [6,7,10,11,17]).


© Carolina Haase, Philipp Kindermann, William Lenhart, and Giuseppe Liotta;
licensed under Creative Commons License CC-BY 4.0



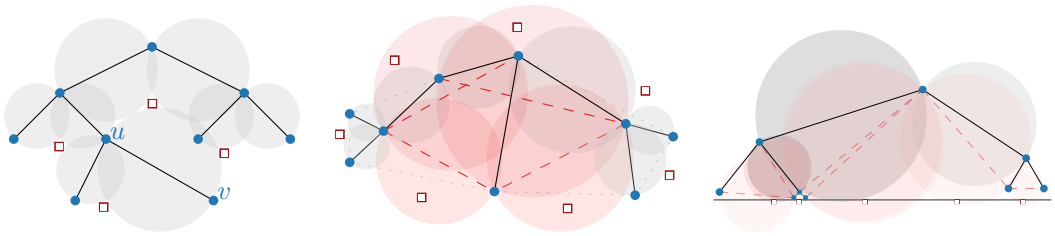


Figure 1 (a) A witness Gabriel drawing, (b) a convexly separable witness Gabriel drawing, and (c) a linearly separable witness Gabriel drawing of a tree. The vertices are drawn as blue filled disks and the witnesses are drawn as red empty squares.

From the graph drawing perspective, a proximity graph is a straight-line drawing where adjacent vertices are close to one another while pairs of non-adjacent vertices are far from each other according to a given definition of closeness. Partly motivated by the applications above, the study of which graph topologies can be realized for some definition of proximity is a well studied question that can be stated as the following graph drawing problem: Given a graph G and a definition of proximity, does G admit a straight-line drawing Γ such that the proximity graph of the vertex set of Γ is isomorphic to G? In the affirmative case, Γ is a *proximity drawing* of G and G is said to be *proximity drawable*. See [15] for a survey on proximity drawings and the proximity drawability problem.

This paper studies *witness proximity drawable* graphs, i.e., graphs that can be realized as the *witness proximity graph* of some point sets. In a witness proximity graph we are given two sets of points: the *vertices*, and the *witnesses* that act as obstacles to the existence of the edges. Any two vertices u and v are adjacent if and only if there is no witness in the neighborhood of u and v. Hence, a *witness Gabriel drawing* of a given graph G is computed by defining a set of points that act as vertices and a set of witnesses that act as obstacles; uv is an edge in the drawing if and only if the Gabriel disk of u and v does not contain any witnesses. Note that the Gabriel disk of an edge uv may contain other vertices and that a witness Gabriel drawing coincides with a Gabriel drawing if the witness set and the vertex set coincide. Figure 1a shows a witness Gabriel drawing Γ of a tree where the Gabriel disk of edge uv contains another vertex of Γ.

Motivated by the design of classifiers in pattern recognition applications, Ichino and Sklansky [12] use witness rectangle of influence graphs[1] to study the correlations among different features, represented by points in the plane. Witness Gabriel graphs are introduced in a seminal paper by Aronov, Dulieu, and Hurtado [2] who also study the witness Gabriel drawability problem for different graph families, including trees and complete bipartite graphs. The same authors extend the results of Ichino and Sklansky about witness rectangle of influence graphs in [3] and study witness Delaunay graphs in [1]. It is worth remarking that both the early work of Ichino and Sklanksy [12] and the subsequent papers of Aronov, Dulieu, and Hurtado [1, 2, 3] stress the importance of the geometric separability between the vertex set and the witness set in a witness proximity graph, since this property provides useful information about the interclass structure of the two point sets.

[1] These graphs are called interclass rectangular influence graphs in [12]. The term witness proximity graph and drawing was first used by Aronov, Hurtado and Dulieu in [1].

This has motivated recent studies about witness Gabriel drawings where the witnesses are either linearly or convexly separable from the vertex set (see, e.g., [9, 13, 14]); the witness set is convexly separable from the vertex set when no witness is in the convex hull of the vertices. At first sight, one might think that the geometric separability of the witness set from the vertex set cannot give rise to sparse topologies. For example, the algorithm of Aronov, Dulieu, and Hurtado [2] to construct witness Gabriel drawings of trees places witnesses well in the interior of the convex hull of the vertex set. However, many sparse topologies are in fact realizable even with linear or convex separability constraints; furthermore, we will show that the geometric separability is actually required for some graph instances that include sparse graphs. Figure 1b is a convexly separable witness Gabriel drawing of the tree of Figure 1a, while Figure 1c is a linearly separable one.

An outline of our results is as follows.

- We show that every outerplanar graph admits a linearly separable witness Gabriel drawing. This extends the result of Aronov, Dulieu, and Hurtado about the witness Gabriel drawability of trees by both enlarging the family of representable graphs and by proving the linear separability of the drawings. The linear separability property for witness Gabriel drawings of outerplanar graphs was previously known only for caterpillars [9].
- We prove that any triangle-free graph has a linearly separable witness Gabriel drawing if and only if it does not have $K_{2,3}$ as a minor; that is, if and only if the triangle-free graph is outerplanar [20]. This extends a result of [13, 14] where it is shown that the complete bipartite graphs admitting linearly separable witness Gabriel drawings are exactly those not containing $K_{2,3}$ as an (induced) subgraph.
- We investigate the relationship between convexly separable and linearly separable witness Gabriel drawable graphs. We construct infinitely many witness Gabriel drawable graphs which admit convexly separable drawings but do not have a linearly separable witness Gabriel drawing. In fact, we show that for any of these graphs, any witness Gabriel drawing must be convexly separable. We believe that identifying graphs whose witness Gabriel drawings require convex separability offers a novel perspective on the long-standing open problem of characterizing witness Gabriel drawable graphs [2].

Our construction of witness Gabriel drawings of outerplanar graphs depends on a new type of drawing which we call a *border parabola drawing* and that may be of interest in its own right. It is a type of drawing in which every vertex is associated with a convex region that contains it and such that there is an edge between two vertices if and only if one end-vertex of an edge is in the region of the other. Any border parabola drawing of a graph can be converted to a linearly separable witness Gabriel drawing with the addition of appropriately chosen witness locations.

2 Preliminaries

We assume familiarity with graph drawing terms and concepts (refer, for example, to [4]). We recall that the Gabriel disk of two distinct points p, q is the disk whose diameter is segment pq. We denote the Gabriel disk of p and q as $\bigcirc[p, q]$. In the rest of the paper $\bigcirc[p, q]$ is a closed set, which is the standard definition of Gabriel disks [8]. We also denote by $\triangle[u, v, w]$ the triangle whose corners are points u, v, and w, and by $\overline{pq}$ the line through points p and q. The following lemma can be proved by elementary geometry.

▶ **Lemma 1.** *Let p, q_1 and q_2 be three non-collinear points and let L be the line through q_1 and q_2. Then the Gabriel disks $\bigcirc[p, q_1]$ and $\bigcirc[p, q_2]$ intersect at a point on L and their union contains all points in $\triangle[p, q_1, q_2]$.*

The lemma has some immediate consequences.

▶ **Corollary 2** ([2] Lemma 6). *Let Γ be a witness Gabriel drawing, let uv and vw be two adjacent edges of Γ. Then: (i) No witness is contained in $\triangle[u,v,w]$, and (ii) any vertex in the interior of $\triangle[u,v,w]$ is adjacent to v in Γ.*

▶ **Corollary 3** ([2] Proposition 1). *Let Γ be a witness Gabriel drawing, and let uv, vw, and uw be edges of Γ forming a three-cycle. Any vertex $r \in \triangle[u,v,w]$ is adjacent to all of $\{u,v,w\}$.*

3 Witness Gabriel Drawings of Outerplanar Graphs

We prove here that every outerplanar graph admits a linearly separable witness Gabriel drawing. The drawings we produce have the additional property that for any edge, its Gabriel disk does not intersect the (open) half-plane containing the witnesses. To do this, we introduce and study a new type of graph drawing, which we call a *border parabola drawing*.

3.1 Border Parabola Drawings

▶ **Definition 4.** *Given a point $p \in \mathbb{R}^2$ with $y(p) > 0$, the* border parabola of p *is the set of points $q \in \mathbb{R}^2$ such that $\bigcirc[p,q]$ is tangent to the x-axis. The border parabola of p can be expressed by the function $f_p(x) = \frac{(x-x(p))^2}{4y(p)}$.*

Note that we use x above both as a variable and as a projection function and that the border parabola of p has p as its focus and $(x(p),0)$ as its vertex. The following property is an immediate consequence of Definition 4.

▶ **Property 1.** *For any points p and q in the upper half-plane:*
- *if $y(p) > f_q(x(p))$, then the Gabriel disk $\bigcirc[p,q]$ lies above the x-axis;*
- *if $y(p) = f_q(x(p))$, then the x-axis is tangent to $\bigcirc[p,q]$;*
- *if $y(p) < f_q(x(p))$, then the Gabriel disk $\bigcirc[p,q]$ crosses the x-axis.*

For any point q in the upper half-plane, let $B_q = \{p : y(p) \geq f_q(x(p))\}$; if $p \in B_q$, we say that p *lies within the border parabola of* q. Property 1 immediately implies that p lies within the border parabola of q if and only if q lies within the border parabola of p, and in particular p is a point of the border parabola of q if and only if q is a point of the border parabola of p.

▶ **Definition 5.** *Given a set P of points in the upper half-plane, the graph $G = (P,E)$, where $E = \{(p,q) : q \in B_p\}$ is called the* border parabola (or PB) graph of P. *The associated straight-line drawing, denoted by $\Gamma(P)$, is called the* border parabola drawing of P.

We say that G admits a *border parabola drawing* if G is the border parabola graph for some set P; if G admits a border parabola drawing, we call G a *border parabola graph* or say that G is *border parabola drawable*. Figure 2 shows a border parabola drawing. The following properties will be useful:

▶ **Property 2.** *Let $\Gamma(P)$ be a border parabola drawing and $\{p_1,p_2,p_3\} \subset P$ be such that $x(p_1) < x(p_2) < x(p_3)$. If $\{p_1,p_3\}$ forms an edge and p_2 lies above $\overline{p_1p_3}$, then $\{p_1,p_2\}$ and $\{p_2,p_3\}$ also form edges.*

Proof. Since the segment $\overline{p_1p_3}$ is contained in both B_{p_1} and B_{p_3}, the portion of the vertical strip determined by $\{p_1,p_3\}$ that lies above $\overline{p_1p_3}$ is completely contained in $B_{p_1} \cap B_{p_3}$ and so p_2 must lie within both parabolas. Hence $\{p_1,p_2\}$ and $\{p_2,p_3\}$ also form edges. ◀

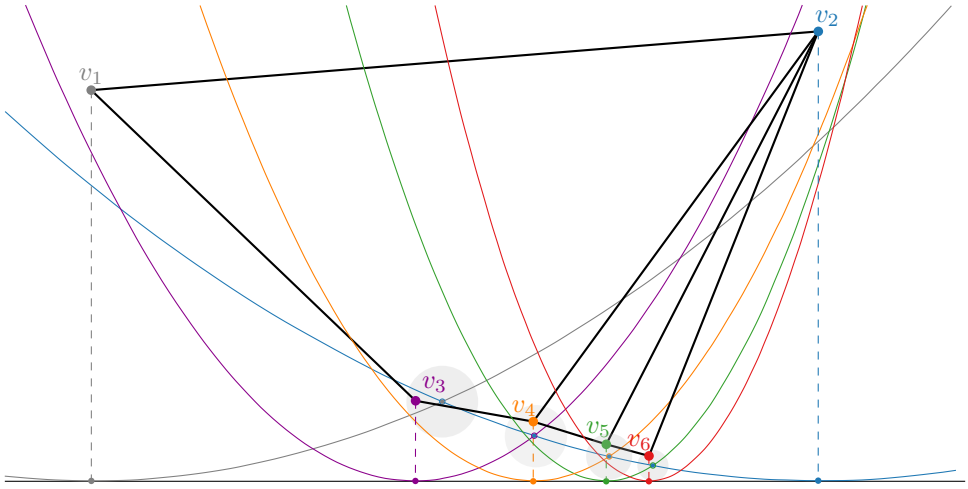


Figure 2 A border parabola drawing of an outerplanar graph.

▶ **Property 3.** *Let $\Gamma(P)$ be a border parabola drawing and let $\{p_0, p_1, p_2\}$ form a triangle in $\Gamma(P)$. Then any point $p \in P \setminus \{p_0, p_1, p_2\}$ contained in the triangle $\triangle(p_0, p_1, p_2)$ is adjacent to each of $\{p_0, p_1, p_2\}$.*

Proof. It suffices to note that $\{p_0, p_1, p_2\}$ all lie in $B_{p_0} \cap B_{p_1} \cap B_{p_2}$, which is a convex set and so contains every point in $\triangle(p_0, p_1, p_2)$. ◀

3.2 Border Parabola Drawings of Outerplanar Graphs

A witness Gabriel drawing $\Gamma(P, W)$ with vertex set P and witness set W of a graph G is *strongly linearly separable* if there is a line separating P from W such that no Gabriel disk of an edge crosses the separating line. We shall say that Γ is a SLSWG drawing of graph G. We assume without loss of generality that the separating line is the x-axis.

Property 1 implies that in a SLSWG drawing $\Gamma(P, W)$, for any $p \in P$, the border parabola of p contains p and its neighbors, but no other point of P. That is, the drawing can be computed from P alone, without reference to W. Hence, we have the following.

▶ **Theorem 6.** *A graph G admits a SLSWG drawing $\Gamma(P, W)$ if and only if G is the border parabola graph of P.*

The question of which graphs admit a SLSWG drawing is therefore the question: Which graphs G are border parabola graphs? We are going to show that outerplanar graphs are border parabola graphs. We start with some definitions.

Let p and q be points in the plane such that $x(p) < x(q)$. The (open) *vertical strip for p and q* is the set $S(p, q) = \{r : x(p) < x(r) < x(q)\}$. The (open) *lower half-strip for p and q* $S^-(p, q)$ is the portion of $S(p, q)$ lying strictly below the line determined by p and q. Note that neither p nor q is contained in $S^-(p, q)$ or $S(p, q)$ because of the strict inequality used in the definitions.

Let $\Gamma(P)$ be a border parabola drawing and let $r \in \mathbb{R}^2$ be any point in the upper half-plane. We define the *neighborhood of r in P* to be the set $N_P(r) = B_r \cap P$; that is $N_P(r)$ is the set of points of P that would be adjacent to r in $\Gamma(P \cup \{r\})$. Finally, for $\{p, q\} \subseteq P$, we say that a subset R of the upper half-plane is an *extension zone for $\{p, q\}$* if

- for all $r \in R$, $N_P(r) \subseteq \{p, q\}$,
- R contains points $r_1, \ldots, r_4$ such that $N_P(r_1) = \emptyset$, $N_P(r_2) = \{p\}$, $N_P(r_3) = \{q\}$, and $N_P(r_4) = \{p, q\}$.

The purpose of an extension zone is to identify, given points $\{p, q\}$, a set of locations such that a point r from the zone could be added to P so that $\Gamma(P \cup \{r\})$ would contain the edges of $\Gamma(P)$ along with both, either, or neither of $\{\{p, r\}, \{q, r\}\}$ but would contain no other additional edges.

Finally, let $\Gamma(P)$ be a border parabola drawing and let $\{p, q\} \subseteq P$. The *meet* of p and q, denoted $m_{p,q}$, is the point of $B_p \cap B_q$ having minimum y-coordinate. The meet of p and q can be equivalently defined to be the point of intersection of f_p and f_q that lies in the vertical strip determined by p and q. Note that the portion of $B_{m_{p,q}}$ below the line determined by p and q is always a subset of $S^-(p, q)$ since p and q both lie on $f_{m_{p,q}}$; thus $m_{p,q} \in S^-(p, q)$.

▶ **Lemma 7.** *Let $\Gamma(P)$ be a border parabola drawing and let $\{p, q\} \subseteq P$ be such that $x(p) < x(q)$, $S^-(p, q) \cap P = \emptyset$ and $N_P(m_{p,q}) = \{p, q\}$. Then there exists a disk D_s centered at $s = m_{p,q}$ such that D_s is an extension zone for $\{p, q\}$, $D_s \subset S^-(p, q)$, and every $r \in D_s$ satisfies*

1. *$S^-(p, r) \cap P = \emptyset$ and $S^-(r, q) \cap P = \emptyset$, and*
2. *$N_{P\cup\{r\}}(m_{p,r}) = \{p, r\}$ and $N_{P\cup\{r\}}(m_{r,q}) = \{r, q\}$,*

In other words, if the pair $\{p, q\}$ satisfies the hypotheses of Lemma 7 for P, then any point r that is close enough to $m_{p,q}$ has the property that both $\{p, r\}$ and $\{r, q\}$ satisfy the hypotheses of Lemma 7 for $P \cup \{r\}$; see Figure 3 for an illustration of the Lemma.

Proof. Since $x(p) < x(q)$ and B_s is a closed region, there exists an $\varepsilon > 0$ such that the disk $D_\varepsilon(s)$ with radius ε and center s has the property that for any $s' \in D_\varepsilon(s)$, $N_P(s') \subseteq N_P(s) = \{p, q\}$. Consider any ε' such that $0 < \varepsilon' \leq \varepsilon$. Since the border parabolas f_p and f_q intersect at s, f_p is strictly increasing at s, and f_q is strictly decreasing at s, the points $p_1 = (x(s), y(s)+\varepsilon')$, $p_2 = (x(s)+\varepsilon', y(s))$, $p_3 = (x(s), y(s)-\varepsilon')$, and $p_4 = (x(s)-\varepsilon', y(s))$ are all in $D_\varepsilon(s)$ and will satisfy $N_P(p_1) = \{p, q\}$, $N_P(p_2) = \{q\}$, $N_P(p_3) = \emptyset$, and $N_P(p_4) = \{p\}$ respectively, so $D_\varepsilon(s)$ is an extension zone for $\{p, q\}$. Also, if $S^-(p, q) \cap P = \emptyset$, then for *any* $r \in S^-(p, q)$, $S^-(p, r) \cap P = \emptyset$ and $S^-(r, q) \cap P = \emptyset$, so the first condition is satisfied.

We now show that every $r \in D_\varepsilon(s)$, satisfies $N_{P\cup\{r\}}(m_{p,r}) = \{p, r\}$; the proof that $N_{P\cup\{r\}}(m_{r,q}) = \{r, q\}$ is analogous. First note that since $S^-(p, q) \cap P = \emptyset$, any point $t \in P \cap N_{P\cup\{r\}}(m_{p,r})$ other than p lies in the portion of $N_{P\cup\{r\}}(m_{p,r})$ above the line L determined by p and q. But that portion of $N_{P\cup\{r\}}(m_{p,r})$ is completely contained in $N_P(s)$, since f_s intersects $f_{m_{p,r}}$ at p and at a point whose x-coordinate is between those of p and r, and between p and r, $f_{m_{p,r}}$ lies below f_s. This, and an analogous argument that $N_{P\cup\{r\}}(m_{r,q}) = \{r, q\}$, establish the second condition. Letting $D_s = D_\varepsilon(s)$ completes the proof. ◀

▶ **Theorem 8.** *Every outerplanar graph $G = (V, E)$ admits a border parabola drawing and hence it has a linearly separable witness Gabriel drawing.*

Proof. We incrementally construct a set P and a function $g : P \to V$ such that at all times the border parabola graph $\Gamma(P)$ of P is isomorphic to the subgraph of G induced by $g(P) \subseteq V$. We assume G is connected; otherwise the construction is applied to each component of G separately, translating successive components far enough to the right of previously drawn components so that if $\{u, v\}$ lie in different components, the Gabriel disk of u and v intersects the negative half-plane.

Assume that we have a topological embedding of G. If G is not a maximal outerplanar graph, add new edges E' to G until $G' = (V, E \cup E')$ is maximal outerplanar. Note that this can be done so that the original embedding of G is preserved. We use the geometric

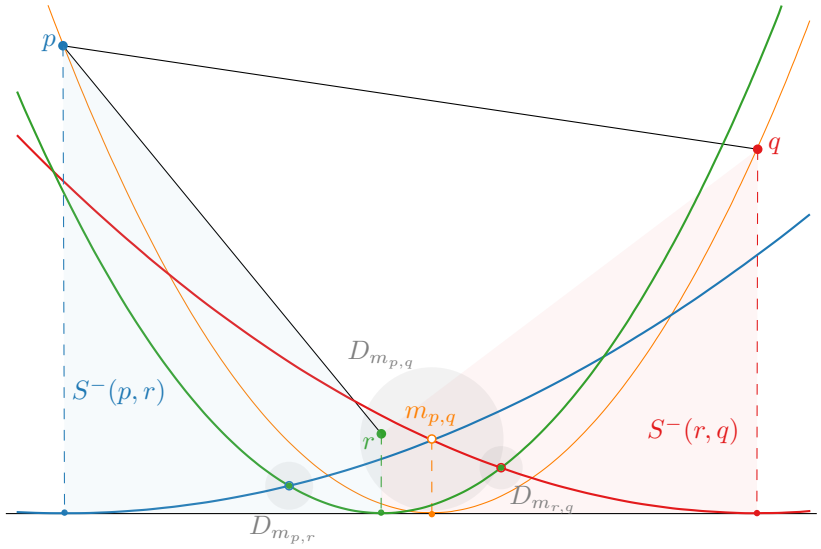


Figure 3 Illustration for Lemma 7.

dual of G' to impose an order on the vertices of G'. Let $\{v_1, v_2\}$ be an edge on the outer face of G' and let v_3 be their common neighbor. Now consider the tree whose vertices are the triangular faces of G' and whose edges connect faces of G' that share an edge. Root this tree at $\{v_1, v_2, v_3\}$. We now order the remaining vertices so that the following invariant holds: For each $i > 3$, v_i forms a triangle with some pair $\{v_j, v_k\}$ such that $\{j, k\}$ are both less than i (note that this also holds for $i = 3$ by construction). Any such ordering will serve our purposes. We call $\{v_j, v_k\}$ the *parents* of v_i.

To begin the construction, if $\{v_1, v_2\} \in E$, let $\{p_1, p_2\}$ be any two points such that $p_2 \in B_{p_1}$; if $\{v_1, v_2\} \in E'$, let $\{p_1, p_2\}$ be any two points such that $p_2 \notin B_{p_1}$ and $m_{p_1,p_2} \in S^-(p_1, p_2)$. In either case, $\{p_1, p_2\}$ can be chosen such that $x(p_1) \neq x(p_2)$, and $y(p_1) \neq y(p_2)$. Let $P = \{p_1, p_2\}$, $g(p_1) = v_1$, and $g(p_2) = v_2$. Note that $\Gamma(P)$ is isomorphic to the subgraph of G (not G'!) induced by $\{v_1, v_2\}$.

At any point in the construction of P, there will be pairs $\{p_j, p_k\} \subset P$ such that $\{v_j, v_k\}$ is an edge of G' and there exists a (unique) m such that $\{v_j, v_k, v_m\}$ is a triangle of G' but p_m has not yet been constructed. We call such a pair $\{p_j, p_k\}$ *pending*.

One other structure that we will take advantage of will be the polygonal chain $C(P)$ consisting of the points in P ordered by increasing x-coordinate, with consecutive pairs forming line segments (note, some of these segments are edges of $\Gamma(P)$ but others may not be); $C(P)$ is a monotone chain with respect to the x-axis. At all times, the next vertex v_i of G' to be associated with a new point p_i in P will have the property that its parents are consecutive points in $C(P)$. After p_i is added to P, the segment between the parents of p_i in $C(P)$ is replaced by segments from p_i to each of its parents. Since it will always be the case that the x-coordinate of p_i is between the x-coordinates of its parents, $C(P)$ will remain monotone at all times.

We now describe how to add points to P so that, for all i with $1 \leq i \leq n$, after point p_i has been added to P, the following invariants hold:

1. $\Gamma(P)$ is isomorphic to the subgraph of G induced by $\{v_1, \ldots, v_i\}$
2. For every j with $3 \leq j \leq i$, p_j is contained in the lower half-strip of its parents
3. The lower half-strips of any two pending pairs in P are disjoint, and

4. For every pending pair $\{p_j, p_k\}$
   a. $\{p_j, p_k\}$ appear as consecutive elements in $C(P)$
   b. $S^-(p_j, p_k) \cap P = \emptyset$,
   c. $m_{p_j,p_k} \in S^-(p_j, p_k)$
   d. $N_P(m_{p_j,p_k}) = \{p_j, p_k\}$

Note that the invariants hold (mostly vacuously) for $i = 1$ and $i = 2$. So assume now that, for some value $i \geq 3$, the invariants hold for $i - 1$. Let $\{v_j, v_k\}$ be the parents of v_i; then $j, k : 1 \leq \{j, k\} \leq i - 1$ and so $\{p_j, p_k\}$ have already been added to P, and so by invariant 4, $\{p_j, p_k\}$ satisfy the hypotheses of Lemma 7. That is, letting $s = m_{p_j,p_k}$, there exists a disk D_s with center s such that D_s is an extension zone for $\{p_j, p_k\}$, $D_s \subset S^-(p_j, p_k)$, and such that every $r \in D_s$ satisfies

1. $S^-(p_j, r) \cap P = \emptyset$ and $S^-(r, p_k) \cap P = \emptyset$.
2. $m_{p_j,r} \in S^-(p_j, r)$ and $m_{r,p_k} \in S^-(r, p_k)$, and
3. $N_{P\cup\{r\}}(m_{p_j,r}) = \{p_j, r\}$ and $N_{P\cup\{r\}}(m_{r,p_k}) = \{r, p_k\}$,

We now choose a location for $p_i \in D_s$ as follows

- If $\{v_j, v_i\} \in E$ and $\{v_k, v_i\} \in E$, choose p_i so that $N_P(p_i) = \{p_j, p_k\}$,
- If $\{v_j, v_i\} \in E$ and $\{v_k, v_i\} \in E'$, choose p_i so that $N_P(p_i) = \{p_j\}$,
- If $\{v_j, v_i\} \in E'$ and $\{v_k, v_i\} \in E$, choose p_i so that $N_P(p_i) = \{p_k\}$,
- If $\{v_j, v_i\} \in E'$ and $\{v_k, v_i\} \in E'$, choose p_i so that $N_P(p_i) = \emptyset$.

Add p_i to P and let $g(p_i) = v_i$, and observe that by virtue of the selection rule for p_i, $\Gamma(P)$ is still isomorphic to the subgraph of G induced by $\{v_1, \ldots, v_i\}$. Also, p_i is contained in the lower half-strip of its parents. In addition, since $S^-(p_j, p_i)$ and $S^-(p_i, p_k)$ are disjoint and contained in $S^-(p_j, p_k)$, and $\{p_j, p_k\}$ is no longer pending, the lower half-strips of all pending pairs are still disjoint, since p_i can't form a pending pair with any point of P other than its parents p_j and p_k (the only neighbors in G' of v_i in $\{v_1, \ldots, v_{i-1}\}$ are v_j and v_k). We thus just need to verify that conditions 4a-4d are still satisfied.

4a Before p_i was added, Condition 4a held. After p_i is added, $\{p_j, p_k\}$ is no longer a pending pair so $\{p_j, p_k\}$ no longer need to be consecutive in $C(P)$. The only possible new pending pairs are $\{p_j, p_i\}$ and $\{p_i, p_k\}$; both of these pairs occur as consecutive points in $C(P)$. and so Condition 4a is satisfied.

4b Similarly, Condition 4b is clearly satisfied since p_i was added inside $S^-(p_j, p_k)$ and so after it has been added to P, p_i is not contained in the lower vertical strip of any pending pair since all such pairs have disjoint strips.

4c Condition 4c is still satisfied for any pending pair not including p_i, so it only needs to be checked for $S^-(p_j, p_i)$ and $S^-(p_i, p_k)$ (which may or may not be pending pairs). But the choice of location of p_i along with Lemma 7 guarantee that m_{p_j,p_i} and m_{p_i,p_k} are in $S^-(p_j, p_i)$ and $S^-(p_i, p_k)$ respectively.

4d Assume that Condition 4d was satisfied before the addition of p_i to P. We need to ensure that, after p_i is added to P, the condition still holds. Note that p_i is selected to lie below $C(P)$. As noted earlier, for any pair of points $\{p, q\} \in P$, the portion of $B_{m_{p,q}}$ below the line determined by $\{p, q\}$ is completely contained in $S^-(p, q)$. So for each pending pair $\{p, q\}$ in $C(P)$, the portion of $B_{m_{p,q}}$ below $C(P)$ is disjoint from the portion of $B_{m_{p',q'}}$ below $C(P)$ for any other pending pair $\{p', q'\}$ in $C(P)$. Thus when p_i is added to P, it will not lie in $B_{m_{p,q}}$ for any pending pair $\{p, q\}$ (other than, perhaps, $\{p_j, p_i\}$ or $\{p_i, p_k\}$), and so the neighborhoods for those pending pairs are unchanged. But by Lemma 7, $N_P(m_{p_j,p_i}) = \{p_j, p_i\}$ and $N_P(m_{p_i,p_k}) = \{p_i, p_k\}$, so Condition 4d is satisfied. ◀

Figure 2 depicts a border parabola drawing constructed with the procedure described in the proof of Theorem 8.

It is worth remarking that Theorem 8 extends a previous result by Aronov, Dulieu, and Hurtado [2] about the witness Gabriel drawability of trees. We also note that the witness Gabriel drawings of trees computed with the algorithm in [2] are neither linearly nor convexly separable.

4 Linearly Separable Triangle-free Witness Gabriel Drawings

As proved in the previous section, any graph that admits a border parabola drawing also admits a linearly separable witness Gabriel Drawing. In this section we will show that, for the family of triangle-free graphs, the converse is also true.

▶ **Lemma 9.** *Let $\Gamma(P, W)$ be a witness Gabriel drawing, and let $\{p_1, p_2, q_1, q_2\} \subseteq P$ be four distinct points such that*

- *$\{p_1, p_2\}$ and $\{q_1, q_2\}$ are crossing edges of $\Gamma(P, W)$, and*
- *$\{p_1, q_1\}$ and $\{p_2, q_2\}$ are non-edges of $\Gamma(P, W)$.*

Then for any witnesses $\{w_1, w_2\} \subseteq W$ for $\{p_1, q_1\}$ and $\{p_2, q_2\}$ respectively, the segment $\overline{w_1 w_2}$ crosses edges $\{p_1, p_2\}$ and $\{q_1, q_2\}$.

Proof. Note that for the hypotheses of the lemma to be satisfied, it cannot be that an endpoint of one of the two edges lies on the other edge, since then the Gabriel disk of one of the non-edges would lie inside the Gabriel disk of one of the edges, making witness placement impossible. Thus the intersection point of the edges is in the interior of each edge.

Let L_1 be the line through $\{p_1, q_2\}$, L_2 be the line through $\{p_2, q_1\}$, and, if the lines are not parallel, let c be their intersection point. By applying a suitable rotation, translation, and/or reflection, we can assume, w.l.o.g., that

- L_1 and L_2 are parallel with slope 0 and are equidistant from the x-axis, or c lies at the origin and L_1 and L_2 have equal but oppositely signed slopes with all of $\{p_1, p_2, q_1, q_2\}$ appearing to the left of c,
- $\{p_1, p_2, q_1, q_2\}$ all lie on or above L_1,
- p_1 lies to the left of q_2 on L_1, and so q_1 lies to the left of p_2 on L_2,

Let $\bigcirc[p_1, p_2]$, $\bigcirc[q_1, q_2]$, $\bigcirc[p_1, q_1]$, and $\bigcirc[p_2, q_2]$ be the Gabriel disks of the two edges and two non-edges. By Lemma 1, each of $\bigcirc[p_1, p_2]$, $\bigcirc[q_1, q_2]$ intersect each of $\bigcirc[p_1, q_1]$, $\bigcirc[p_2, q_2]$ at two points and all eight such points of intersection lie on L_1 and L_2. Four of the eight points are $\{p_1, p_2, q_1, q_2\}$; denote the other four by $\{P_1, P_2, Q_1, Q_2\}$ as follows: If $r \in \{p_1, p_2, q_1, q_2\}$, then R denotes the other point of intersection of the two Gabriel disks that intersect at r. Note that if $r \in \{p_1, p_2, q_1, q_2\}$ lies on L_i, for $1 = 1, 2$, then R lies on L_{3-i}. Figure 4 and 5 show the two possible configurations.

Note that since p_1 is to the left of q_2 on L_1, P_1 must be to the left of Q_2 on L_2; similarly Q_1 must be to the left of P_2 on L_1, since q_1 is to the left of p_2 on L_2.

Consider first the case where L_1 and L_2 are parallel and consider L_2 and the region above it. Now consider the portions of the Gabriel disks for the two edges that lie above L_2. One portion connects q_1 to Q_2 and the other connects p_2 to P_1. But since q_1 must be to the left of p_2 and P_1 must be to the left of Q_2, the projections of those portions onto L_2 intersect and so either the portions cross or one is contained below the other. Since all of the eight intersection points of the four Gabriel disks lie on L_1 and L_2 either every point on the portions of the Gabriel disks of the two non-edges lying above L_2 lie below some point on a portion of one (or both) of the Gabriel disks for the two edges, or none do. But near

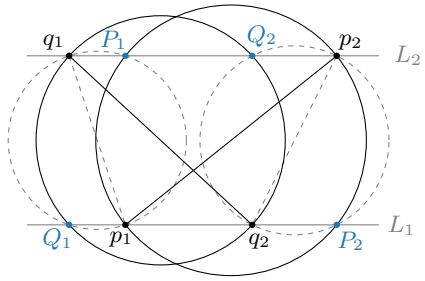


Figure 4 Illustration for Lemma 9 – parallel case.

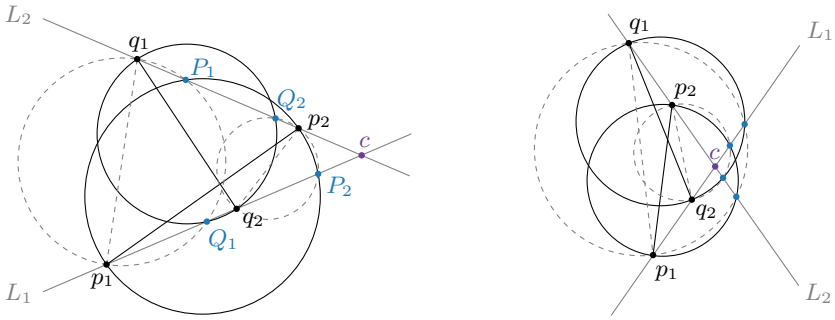


Figure 5 Illustration for Lemma 9 – skew case.

the left-most of the four intersection points on L_2 (q_1 in Figure 4), the portion of $\bigcirc[q_1, q_2]$ above L_2 contains the portion of $\bigcirc[q_1, p_1]$ above L_2. Similarly, near the right-most of the four intersection points on L_2 (p_2 in Figure 4), the portion of $\bigcirc[p_2, p_1]$ above L_2 contains the portion of $\bigcirc[p_2, q_2]$ above L_2. Thus no portion of either of the Gabriel disks for the non-edges lies both above L_2 and outside the union of the Gabriel disks for the two edges.

A similar argument holds for the region below L_1. Therefore any witnesses for $\bigcirc[p_1, q_1]$ and $\bigcirc[p_2, q_2]$ must lie in the horizontal strip between L_1 and L_2. Note now that the entire quadrilateral bounded by $\{p_1, p_2, q_1, q_2\}$ is contained in the union of the Gabriel disks of the two edges. Note also that the only points to the right of that quadrilateral and between L_1 and L_2 that disk $\bigcirc[q_1, p_1]$ can contain are also contained in $\bigcirc[p_1, p_2] \cup \bigcirc[q_1, q_2]$: for any point p' that lies in this region, the angle $q_1p'q_2$ is larger than the angle $q_1p'p_1$, so if p' lies in $\bigcirc[q_1, p_1]$, then it also lies in $\bigcirc[q_1q_2]$. Similarly, the only points to the left of that quadrilateral and between L_1 and L_2 that disk $\bigcirc[p_2, q_2]$ can contain are also contained in $\bigcirc[p_1, p_2] \cup \bigcirc[q_1, q_2]$. Thus a witness for $\bigcirc[q_1, p_1]$ must lie in the strip between L_1 and L_2 to the left of the quadrilateral and a witness for $\bigcirc[q_2, p_2]$ must lie in the strip between L_1 and L_2 to the right of the quadrilateral, implying that the segment between them crosses both $\{p_1, p_2\}$ and $\{q_1, q_2\}$.

The proof in the case that L_1 and L_2 intersect is similar but slightly more involved. Instead of three regions of interest (below L_1, between L_1 and L_2, and above L_2), there are now four regions that we call the *north, east, south, and west regions*:

- north: the intersection of the regions above both L_1 and L_2,
- south: the intersection of the regions below both L_1 and L_2,
- west: the intersection of the region above L_1 and below L_2, and
- east: the intersection of the region below L_1 and above L_2.

Note that by construction, $\{p_1, p_2, q_1, q_2\}$ are contained in the west region.

The analogous proof for this case uses the following observations:

- Any point within either of the Gabriel disks for the two non-edges that lies in the north or south regions is also within the union of the Gabriel disks for the two edges. Thus any witnesses for the two non-edges must lie in the west and/or east regions.
- The Gabriel disks of the two edges completely contain the quadrilateral formed by $\{p_1, p_2, q_1, q_2\}$,
- The only points to the left of that quadrilateral and within the west region that disk $\bigcirc[q_2, p_2]$ can contain are also contained in $\bigcirc[p_1, p_2] \cup \bigcirc[q_1, q_2]$.

However, it is not always the case that the disk $\bigcirc[p_1, q_1]$ does not contain any points to the right of that quadrilateral. There are two situations, depending upon whether c, the intersection point of L_1 and L_2, lies inside the four Gabriel disks or not. (It either lies in all four of the disks if $\angle q_1 c q_2 = \angle p_1 c p_2 \geq \pi/2$ or in none of them if not.) If c lies outside of the four disks, then, in fact, the only points to the right of that quadrilateral and within the west region that disk $\bigcirc[p_1, q_1]$ can contain are also contained in $\bigcirc[p_1, p_2] \cup \bigcirc[q_1, q_2]$. Moreover, in this case, neither of the Gabriel disks of the non-edges intersects the east region, and so witnesses for the two non-edges must both lie in the west region, but on opposite sides of the quadrilateral. Thus the segment between them will cross both $\{p_1, p_2\}$ and $\{q_1, q_2\}$.

If c lies inside the four Gabriel disks, then the portion of $\bigcirc[q_2, p_2]$ that lies in the east region (and, in fact, all of $\bigcirc[q_2, p_2]$) will be contained in the union of the Gabriel disks of the two edges, so there will be no location at which a witness for $G[q_2, p_2]$ can be placed. ◀

As a final ingredient, we will prove the following generalization of Property 2:

▶ **Property 4.** *Let $\Gamma(P)$ be a linearly separable witness Gabriel drawing and $\{p_1, p_2, p_3\} \subset P$ be such that $x(p_1) < x(p_2) < x(p_3)$. If $\{p_1, p_3\}$ forms an edge and p_2 lies above $\overline{p_1 p_3}$, then $\{p_1, p_2\}$ and $\{p_2, p_3\}$ also form edges.*

Proof. It suffices to note that the portion of the Gabriel disk $\bigcirc[p_1, p_3]$ below segment $\overline{p_1 p_3}$ contains the portions of the Gabriel disks $\bigcirc[p_1, p_2]$ and $\bigcirc[p_2, p_3]$ that lie below $\overline{p_1 p_3}$. ◀

We are now ready to state the main result of this section.

▶ **Theorem 10.** *The following statements are equivalent for a triangle-free graph G:*

- *G is outerplanar*
- *G admits a linearly separable witness Gabriel drawing*
- *G admits a border parabola drawing.*

Proof. By Theorem 8, every outerplanar graph admits a border parabola drawing, and by definition every border parabola drawing is also a linearly separable witness Gabriel drawing. To show the equivalency, we still have to argue that, if a triangle-free graph admits a linearly separable witness Gabriel drawing, then it must be outerplanar.

An immediate consequence of Lemma 9 is that if $\Gamma(P, W)$ is a linearly separable witness Gabriel drawing, then P cannot contain four points satisfying the hypotheses of the lemma: P and W are contained in disjoint half-planes and so no segment between two points in P can cross a segment between two points in W. In particular, if $\Gamma(P, W)$ is a linearly separable witness Gabriel drawing of a triangle-free graph, then $\Gamma(P, W)$ must be a planar drawing, since any pair of crossing edges would satisfy the hypotheses of Lemma 9. Moreover, no point of P in such a drawing could lie vertically above any edge, since, otherwise, by Property 4, the drawing would contain a triangle, and so the drawing must, in fact, be outerplanar. ◀

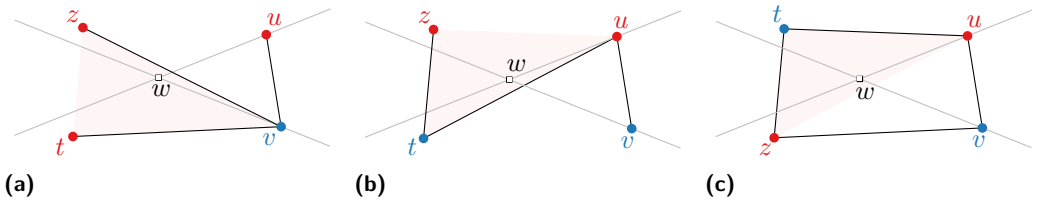


Figure 6 Illustration for the proof of Lemma 11.

We recall that results of [13,14] show that the complete bipartite graphs admitting linearly separable witness Gabriel drawings are exactly those not containing $K_{2,3}$ as an induced subgraph. Following a characterization of triangle-free outerplanar graphs by Syslo [20], Theorem 10 extends these results showing that any triangle-free graph has a linearly separable witness Gabriel drawing if and only if it does not have $K_{2,3}$ as a minor.

5 Convexly Separable Witness Gabriel Drawings

A witness Gabriel drawing Γ is said to be convexly separable if none of its witnesses lie within the interior of the convex hull of Γ. Clearly, if a family of graphs admits linearly separable witness Gabriel drawings, then it also admits convexly separable ones. This holds, for example, for outerplanar graphs (see Theorem 8). However, two natural questions arise: (i) If a graph admits a convexly separable witness Gabriel drawing, does it always admit a linearly separable one? (ii) Are there graphs that are witness Gabriel drawable, but for which all such drawings must be convexly separable?

In this section, we answer the first question negatively and the second affirmatively by constructing witness Gabriel drawable graphs that require convex separability in any witness Gabriel drawing, but for which linear separability is not possible.

▶ **Lemma 11.** *Every $K_{m,n}$ such that $m \geq 2$ and $n \geq 2$ admits a witness Gabriel drawing. Also, if $n > 2$ any such drawing must be convexly separable but cannot be linearly separable.*

Proof. Every $K_{m,n}$ is witness Gabriel drawable by Theorem 5 of [2]. Also, for any $n > 2$ $K_{m,n}$ contains $K_{2,3}$ as a subgraph and hence, by Theorem 10, it does not admit a linearly separable witness Gabriel drawing. It remains to show that every witness Gabriel drawing of $K_{m,n}$ is convexly separable. We color red every vertex of one partition set and blue every vertex of the other partition set of $K_{m,n}$.

Let Γ be a witness Gabriel drawing of $K_{m,n}$ and assume that there is a witness w in the interior of the convex hull and let uv be an edge of $K_{m,n}$ such that u is a red vertex and v is a blue vertex. Consider the line L_1 passing through u and w and the line L_2 passing through v and w. Lines L_1 and L_2 divide the plane into four wedges each having w as their apex. We call these wedges the *top wedge*, *bottom wedge*, *left wedge*, and *right wedge* of w, respectively. The top (left) wedge of w is opposite to its bottom (right) wedge. We assume without loss of generality that edge uv is in the right wedge of w. See also Figure 6.

Observe that no vertex z of Γ can be in the left wedge of w. Namely, depending on its color, vertex z would be adjacent to either u or v and we would have that w is a point of the triangle whose corners are u, v, and z, violating Corollary 2. Also, no edge zt of the convex hull of Γ can cross the left wedge of w: If z and t have the same color, let s be whichever of u and v has the other color. Now $\triangle[z,t,s]$ contains w, violating Corollary 2. If z and t have different colors, they form an edge of Γ and $\triangle[z,t,u]$ contains w, which again violates Corollary 2.

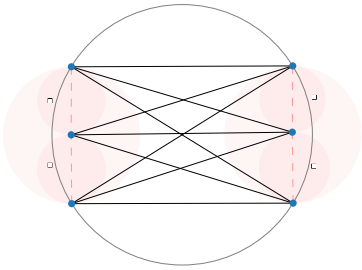


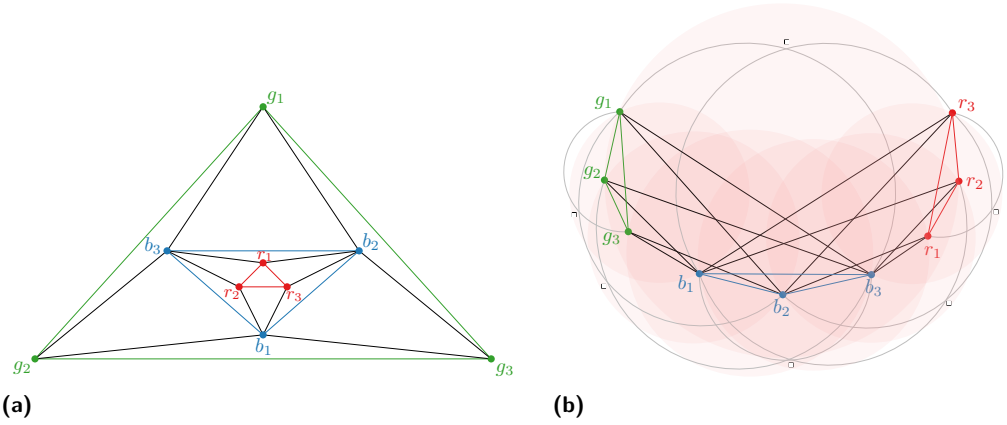
Figure 7 A witness Gabriel drawing of $K_{3,3}$.

Since no vertex and no edge of the convex hull of Γ can intersect the left wedge of w, it follows that w is not in the interior of the convex hull of Γ. ◀

Figure 7 shows a convexly separable witness Gabriel drawing of $K_{3,3}$.

The graphs of Lemma 11 are triangle-free and dense. We now prove a similar result for a planar triangulation consisting of just nine vertices.

Let $C = \{r, g, b\}$ and $I = \{1, 2, 3\}$. Then we define the graph G_9 to have vertex set $V = \{c_i : c \in C, i \in I\}$ and edge set $E = \{\{r_i, r_j\}, \{b_i, b_j\}, \{g_i, g_j\}, \{b_i, r_j\}, \{b_i, g_j\} : 1 \leq i, j \leq 3, i \neq j\}$. Note that G_9 can also be described as the complete graph on V minus all edges $\{r_i, g_j\}, i, j \in I$ and $\{c_i, c'_i\}, c, c' \in C, i \in I$; see Figure 8a. One can also obtain the graph G_9 by taking two octahedra and gluing them together at a triangular face. The vertex $v = c_i \in V$, $c \in C, i \in I$, is said to have *color* $c(v) = c$ and *index* $i(v) = i$; the *index set* of $V' \subseteq V$ is given by $I(V') = \{i(v) : v \in V'\}$. We will refer to vertices having color r as *red* vertices, color b as *blue* vertices, and color g as *green* vertices.



(a) **(b)**

Figure 8 Graph G_9 and a convexly separable witness Gabriel drawing of G_9.

By means of Lemma 9, Corollary 2 and Corollary 3, we can now prove the following.

▶ **Lemma 12.** *Graph G_9 admits a witness Gabriel drawing. Also any such drawing must be convexly separable but cannot be linearly separable. Any proper subgraph of G_9 admits a linearly separable witness Gabriel drawing.*

Proof. It is possible to construct a convexly separable witness Gabriel drawing of G_9. See Figure 8b for such an example.

We now show that no witness Gabriel drawing of G_9 can be linearly separable. Assume that there existed a linearly separable witness Gabriel Drawing Γ of G_9. By Lemma 9, Corollary 3, and Corollary 2, Γ must be such that: (1) Any $V' \subset V$ having $|I(V')| < |I| = 3$



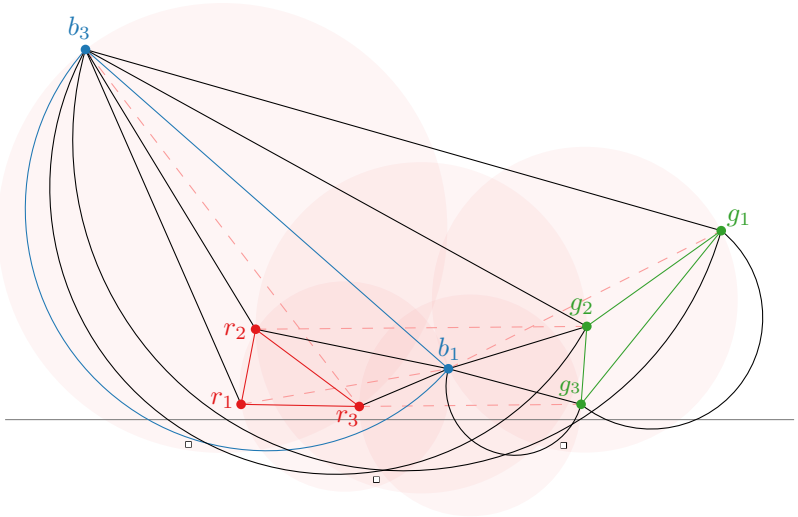
Figure 9 Illustration for the proof of Lemma 12: A witness Gabriel drawing Γ of G_9 must be convexly separable.

induces a plane subgraph of Γ. Note that since V' does not contain vertices of all three indices, the subgraph induced by V' contains no triangles. Thus, by Lemma 9, no two of its edges can cross. (2) Any induced 4-cycle of Γ must form a convex quadrilateral: By (1), the 4-cycle forms a plane subgraph of Γ. If it were not convex, one of the four vertices would be contained in the wedge formed by the other three and so, by Corollary 2, a diagonal would exist. (3) For any $\{i, j\} \subset I$, the subgraph $G_{i,j}$ of G_9 induced by all vertices having indices in $\{i, j\}$ is a plane subgraph of Γ consisting of the two convex quadrilaterals $R_{i,j} = \{b_i, b_j, r_i, r_j\}$ and $G_{i,j} = \{b_i, b_j, g_i, g_j\}$. Further, these two quadrilaterals lie on opposite sides of the line $L_{i,j}$ determined by $\{b_i, b_j\}$. Both $R_{i,j}$ and $G_{i,j}$, which we will call the *red* and *green* quadrilaterals, respectively, are induced 4-cycles of G_9, and so by (2), each forms a convex quadrilateral in Γ. If both were on the same side of $L_{i,j}$, then either some edge of $R_{i,j}$ intersects some edge of $G_{i,j}$, which is forbidden by (1), or one of the two quadrilaterals is contained in the other. Suppose that $R_{i,j}$ contains $G_{i,j}$; in particular $R_{i,j}$ contains the vertex g_i. Now b_k $(k \neq i, j)$ is adjacent to all of $\{b_i, b_j, r_i, r_j\}$ and the triangles $\triangle(b_k, b_i, b_j)$, $\triangle(b_k, b_j, r_i)$, $\triangle(b_k, r_i, r_j)$, and $\triangle(b_k, r_j, b_i)$ completely cover $R_{i,j}$. This means, however, that one of the four triangles contains g_i, violating Corollary 3, thus $R_{i,j}$ and $G_{i,j}$ must lie on opposite sides of $L_{i,j}$.

Now consider the triangle $\triangle = \triangle(b_1, b_2, b_3)$ and the three lines $L_{1,2}, L_{2,3}, L_{3,1}$ determined by it's edges. We will refer to the half-plane of $L_{i,j}$ *not* containing $\triangle$ as the *outer half-plane* of $L_{i,j}$. Since each outer half-plane $L_{i,j}$ contains one of $R_{i,j}$ and $G_{i,j}$, then by the pigeon-hole principle, there are either at least two red quadrilaterals or two green quadrilaterals contained in their respective outer half-planes. So assume that, w.l.o.g., $R_{i,j}$ and $R_{i,k}$ are contained in the outer half-planes of $L_{i,j}$ and $L_{i,k}$, respectively. Then, in particular, r_i is contained in the intersection of these two outer half-planes. But this implies that b_i is contained in the triangle $\triangle(r_i, b_j, b_k)$, violating Corollary 3. It follows that no witness Gabriel drawing of G_9 can be linearly separable.

With a similar reasoning as in the proof of Lemma 11, we now show that every witness Gabriel drawing Γ of G_9 must be convexly separable. Suppose that there existed a witness w in the interior of the convex hull of Γ. Consider the smallest infinite wedge with apex w containing the blue vertices of Γ and extend the rays of the wedge to two lines that intersect at w. These two lines define four wedges each of them having w as an apex. We call these wedges the *top wedge*, *bottom wedge*, *left wedge*, and *right wedge* of w, respectively. The top (left) wedge of w is opposite to its bottom (right) wedge. We assume without loss of generality that the three blue vertices lie in the right wedge of w and denote as b_i and b_j the two blue vertices along the boundary of this wedge; see Figure 9 for an illustration.

We show that no edge of the convex hull of Γ can intersect the left wedge of w. First observe that there cannot be a vertex z in the left wedge of w: If vertex z existed it would be adjacent to one of $\{b_i, b_j\}$ and therefore w would a point in the interior of the triangle whose corners are b_i, b_j, and z, which contradicts Corollary 2. Furthermore, no edge zt of



Vertex	g_1	g_2	g_3	b_1	b_3	r_1	r_2	r_3	w_1	w_2	w_3
x	116	93	98	71	8	33	44	51	18	60	95
y	29	15	7	12	74	6	14	3	-1	-8	-2

Figure 10 Coordinates of the vertices and witnesses in a linearly separable witness Gabriel drawing of G_9 minus a blue vertex.

the convex hull of Γ can cross the left wedge of w: Neither z nor t is a blue vertex; also $I(\{z,t\})$ has size at most 2, so there is some blue vertex b adjacent to both z and t. The wedge formed by these three vertices contains w, violating Corollary 2.

Since no edge of the convex hull of Γ can intersect the left wedge of w, it follows that w is not in the interior of the convex hull of Γ.

To conclude the proof, we need to show that every proper subgraph of G_9 has a linearly separable Gabriel drawing. It is sufficient to show that deleting any vertex of G_9 results in a graph that has a linearly separable witness Gabriel drawing. Figures 10 and 11 depict linearly separable witness Gabriel drawings (and exact coordinates) of G_9 minus a green vertex and of G_9 minus a blue vertex, respectively. ◀

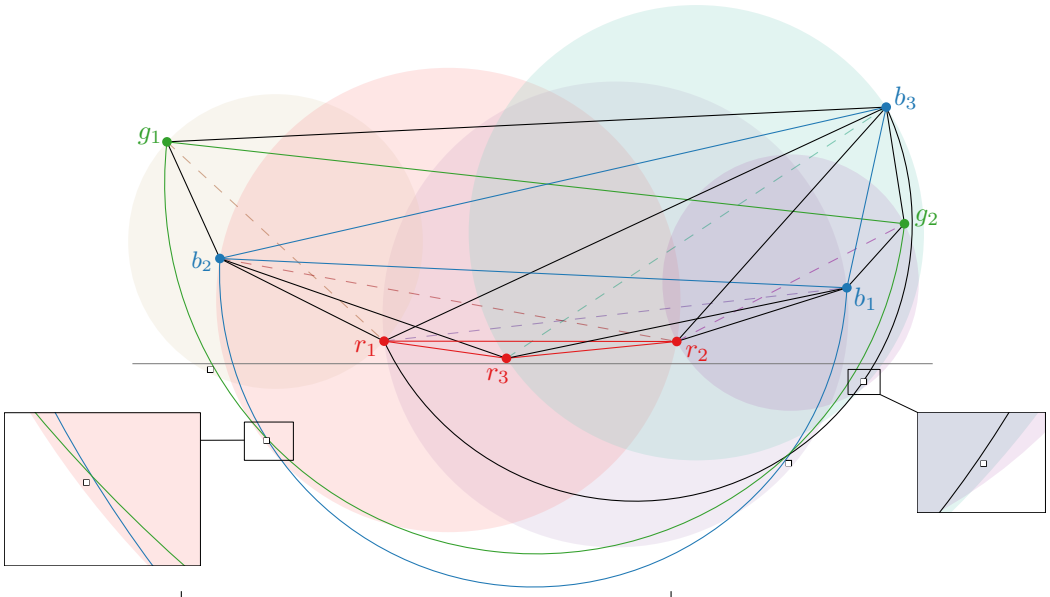
Figure 8b depicts a convexly separable witness Gabriel drawing of G_9, while Figures 10 and 11 depict linearly separable witness Gabriel drawings and exact coordinate of G_9 minus a green vertex and of G_9 minus a blue vertex, respectively. Lemma 12 complements Lemma 11 by providing an example which is a triangulated planar graph.

If we denote as $\mathcal{CSWGD}$ the set of those witness Gabriel drawable graphs whose witness Gabriel drawings must be convexly separable, and as $\mathcal{LSWGD}$ the set of those witness Gabriel drawable graphs that admit a linearly separable witness Gabriel drawing, then the results of this section can be summarized as follows.

▶ **Theorem 13.** *$\mathcal{LSWGD} \subsetneq \mathcal{CSWGD}$. Also, $\mathcal{CSWGD} \setminus \mathcal{LSWGD}$ contains both bipartite graphs and planar triangulations.*

6 Concluding Remarks and Open Problems

In this paper we have studied witness Gabriel drawings where the sets of vertices and the set of witnesses are either linearly or convexly separable. We showed that every outerplanar graph admits a linearly separable witness Gabriel drawing. We remark that the witness



Vertex	g_1	g_2	b_1	b_2	b_3	r_1	r_2	r_3	w_1	w_2	w_3	w_4
x	0	126	116	9	123	37	87	58	8	17	106	119
y	38	24	113	18	44	4	4	1	-1	-13	-17	-3

Figure 11 Drawing and exact coordinates of the vertices and witnesses in a linearly separable witness Gabriel drawing of G_9 minus a green vertex.

Gabriel drawability of outerplanar graphs was known only for trees and, in addition, the known construction was not giving rise to separable realizations. We then characterized the triangle-free graphs that admit a linearly separable witness Gabriel drawing. Finally we proved that the family of linearly separable witness Gabriel drawable graphs is a proper subset of the convexly separable ones.

Several open problems naturally arise from our results. We conclude the paper by listing some of those that, in our opinion, are among the most interesting.

- Further investigate the relationship between border parabola graphs and linearly separable witness Gabriel drawable graphs. Specifically, are there graphs that admit a linearly separable Gabriel drawing but do not have a border parabola drawing?
- Extend the characterization of Theorem 10 to graphs that have 3-cycles.
- Characterize the convexly separable witness Gabriel drawings.
- Finally, an immediate consequence of Lemma 12 is that there are two planar triangulations (namely two copies of G_9) which do not admit a mutual witness Gabriel drawing. A mutual witness Gabriel drawing consists of a pair of witness Gabriel drawings where the vertex set of one drawing acts as the witness set for the other and vice versa. Clearly, if every witness Gabriel drawing of G_9 is convexly separable, any mutual witness Gabriel drawing of two copies of G_9 should be linearly separable, which is however impossible by Lemma 12. It would be interesting to further study pairs of planar graphs that admit or do not admit a mutual witness Gabriel drawing. Results in this direction are known for pairs of isomorphic trees and for bipartite graphs [9, 13, 14].

References

1 Boris Aronov, Muriel Dulieu, and Ferran Hurtado. Witness (Delaunay) graphs. *Comput. Geom.*, 44(6-7):329–344, 2011. doi:10.1016/j.comgeo.2011.01.001.

2 Boris Aronov, Muriel Dulieu, and Ferran Hurtado. Witness Gabriel graphs. *Comput. Geom.*, 46(7):894–908, 2013. doi:10.1016/J.COMGEO.2011.06.004.

3 Boris Aronov, Muriel Dulieu, and Ferran Hurtado. Witness rectangle graphs. *Graphs Comb.*, 30(4):827–846, 2014. doi:10.1007/s00373-013-1316-x.

4 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.

5 Miguel Á. Carreira-Perpiñán and Richard S. Zemel. Proximity graphs for clustering and manifold learning. In L. Saul, Y. Weiss, and L. Bottou, editors, *17th Annual Conference on Neural Information Processing Systems (NIPS 2004)*, volume 17, pages 225–232. MIT Press, 2004. URL: https://proceedings.neurips.cc/paper/2004/hash/dcda54e29207294d8e7e1b537338b1c0-Abstract.html.

6 Peter Eades, Seok-Hee Hong, Karsten Klein, and An Nguyen. Shape-based quality metrics for large graph visualization. In Emilio Di Giacomo and Anna Lubiw, editors, *23rd International Symposium on Graph Drawing and Network Visualization (GD 2015)*, volume 9411 of *LNCS*, pages 502–514. Springer, 2015. doi:10.1007/978-3-319-27261-0_41.

7 Peter Eades, Seok-Hee Hong, An Nguyen, and Karsten Klein. Shape-based quality metrics for large graph visualization. *J. Graph Algorithms Appl.*, 21(1):29–53, 2017. doi:10.7155/jgaa.00405.

8 K. R. Gabriel and R. R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969. doi:10.2307/2412323.

9 Carolina Haase, Philipp Kindermann, William J. Lenhart, and Giuseppe Liotta. Mutual witness proximity drawings of isomorphic trees. In Michael A. Bekos and Markus Chimani, editors, *31st International Symposium on Graph Drawing and Network Visualization (GD 2023)*, volume 14465 of *LNCS*, pages 304–319. Springer, 2023. doi:10.1007/978-3-031-49272-3_21.

10 Seok-Hee Hong and Patrick Eades. β-skeleton shape-based metrics for large and complex graph drawings. In *17th IEEE Pacific Visualization Conference (PacificVis 2024)*, pages 277–282. IEEE, 2024. doi:10.1109/PACIFICVIS60374.2024.00038.

11 Seok-Hee Hong, Amyra Meidiana, James Wood, Juan Pablo Ataides, Peter Eades, and Kunsoo Park. dGG, dRNG, DSC: New degree-based shape-based faithfulness metrics for large and complex graph visualization. In *15th IEEE Pacific Visualization Symposium (PacificVis 2022)*, pages 51–60. IEEE, 2022. doi:10.1109/PACIFICVIS53943.2022.00014.

12 Manabu Ichino and Jack Sklansky. The relative neighborhood graph for mixed feature variables. *Pattern Recognit.*, 18(2):161–167, 1985. doi:10.1016/0031-3203(85)90040-8.

13 William J. Lenhart and Giuseppe Liotta. Mutual witness Gabriel drawings of complete bipartite graphs. In Patrizio Angelini and Reinhard von Hanxleden, editors, *30th International Symposium on Graph Drawing and Network Visualization (GD 2022)*, volume 13764 of *LNCS*, pages 25–39. Springer, 2022. doi:10.1007/978-3-031-22203-0_3.

14 William J. Lenhart and Giuseppe Liotta. Mutual witness Gabriel drawings of complete bipartite graphs. *Theor. Comput. Sci.*, 974:114115, 2023. doi:10.1016/J.TCS.2023.114115.

15 Giuseppe Liotta. Proximity drawings. In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization*, pages 115–154. Chapman and Hall/CRC, 2013. URL: https://cs.brown.edu/people/rtamassi/gdhandbook/chapters/proximity.pdf.

16 Luke Mathieson and Pablo Moscato. *An Introduction to Proximity Graphs*, pages 213–233. Springer International Publishing, Cham, 2019. doi:10.1007/978-3-030-06222-4_4.

17 Amyra Meidiana, Seok-Hee Hong, and Peter Eades. Shape-faithful graph drawings. In Patrizio Angelini and Reinhard von Hanxleden, editors, *30th International Symposium on Graph Drawing and Network Visualization (GD 2022)*, volume 13764 of *LNCS*, pages 93–108. Springer, 2022. doi:10.1007/978-3-031-22203-0_8.

18 Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, Second Edition*. Wiley Series in Probability and Mathematical Statistics. Wiley, 2000. doi:10.1002/9780470317013.

19 Joseph O'Rourke and Godfried T. Toussaint. Pattern recognition. In Jacob E. Goodman, Joseph O'Rourke, and Csaba Toth, editors, *Handbook of Discrete and Computational Geometry, Third Edition*. Chapman and Hall/CRC, 2017. URL: http://www.csun.edu/~ctoth/Handbook/chap54.pdf.

20 Maciej M. Syslo. Characterizations of outerplanar graphs. *Discret. Math.*, 26(1):47–53, 1979. doi:10.1016/0012-365X(79)90060-8.

21 Godfried T. Toussaint and Constantin Berzan. Proximity-graph instance-based learning, support vector machines, and high dimensionality: An empirical comparison. In Petra Perner, editor, *8th International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM 2012)*, volume 7376 of *LNCS*, pages 222–236. Springer, 2012. doi:10.1007/978-3-642-31537-4_18.

OOPS: Optimized One-Planarity Solver via SAT

Sergey Pupyrev
Menlo Park, CA, USA

Abstract

We present OOPS (Optimized One-Planarity Solver), a practical heuristic for recognizing 1-planar graphs and several important subclasses. A graph is 1-planar if it can be drawn in the plane such that each edge is crossed at most once – a natural generalization of planar graphs that has received increasing attention in graph drawing and beyond-planar graph theory. Although testing planarity can be done in linear time, recognizing 1-planar graphs is NP-complete, making effective practical algorithms especially valuable.

The core idea of our approach is to reduce the recognition of 1-planarity to a propositional satisfiability (SAT) instance, enabling the use of modern SAT solvers to efficiently explore the search space. Despite the inherent complexity of the problem, our method is substantially faster in practice than naïve or brute-force algorithms. In addition to demonstrating the empirical performance of our solver on synthetic and real-world instances, we show how OOPS can be used as a discovery tool in theoretical graph theory. Specifically, we employ OOPS to investigate two research problems concerning 1-planarity of specific graph families. Our implementation of the algorithm is publicly available to support further exploration in the field.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Mathematics of computing → Graph theory

Keywords and phrases beyond planarity, 1-planar graph, SAT, book embeddings, upward 1-planarity

Digital Object Identifier 10.4230/LIPIcs.GD.2025.14

Supplementary Material *Software (Source Code)*: `https://github.com/spupyrev/oops` [34]

1 Introduction

Planarity is a central concept in graph drawing; it describes graphs that can be embedded in the plane without edge crossings. Thanks to classical results, planarity testing can be carried out in linear time, and planar embeddings can be constructed for very large instances. However, most graphs arising in real-world applications, such as bioinformatics and software engineering, are non-planar. This motivates the study of *beyond-planar* graph classes, which extend planarity by allowing a limited number of edge crossings.

One of the most natural such classes is that of *1-planar* graphs, introduced by Ringel in 1965 in the context of vertex-face colorings of plane graphs [36]. A graph is 1-planar if it can be drawn in the plane such that each edge is crossed at most once. This definition generalizes planar graphs but retains many of their structural properties, such as bounded edge density, bounded chromatic number, small separators, and low treewidth. The class of 1-planar graphs is receiving an increasing attention in the recent years: While the annotated bibliography on 1-planarity by Kobourov, Liotta, and Montecchiani from 2017 [28] lists 143 references on the topic, a recent Google Scholar search of "1-planar graph" reveals several thousand of related publications.

Despite their structural similarities to planar graphs, 1-planar graphs are substantially more challenging from a computational standpoint. In contrast to the linear-time algorithms for planarity testing, recognizing whether a given graph is 1-planar is NP-complete, even for restricted graph families [14, 24, 29]. As a result, research on 1-planarity recognition has focused on developing algorithms for restricted subclasses, such as IC-planar and outer-1-planar graphs [1, 5, 25, 39], or on parameterized and approximation algorithms [8, 20]. While


© Sergey Pupyrev;
licensed under Creative Commons License CC-BY 4.0

these results have deepened our theoretical understanding of 1-planarity, there remains a significant gap between theory and practice: existing exact algorithms are typically not scalable for use on general graphs arising in real-world scenarios. The design and evaluation of practical algorithms for recognizing (subclasses of) 1-planar graphs remains an underexplored direction [10, 28].

1.1 Summary of Contributions

In this paper, we reduce the gap by designing and implementing a practical heuristic for testing 1-planarity. Besides a naïve exhaustive search to enumerate all possible crossings in a graph, we are aware of only one attempt to implement a general algorithm for recognizing 1-planar graphs [10]. While this backtracking algorithm is carefully engineered, it is not efficient enough to answer basic research questions like "What is the smallest bipartite non-1-planar graph?" or "Is the 28-vertex Coxeter graph 1-planar?". In contrast, our new algorithm, which we call `OOPS` (`O`ptimized `O`ne-`P`lanarity `S`olver), provides a substantial speedup and thus capable to answer such questions in seconds.

Our heuristic is based on converting the 1-planarity problem into a propositional satisfiability (SAT) instance, which enables the use of modern SAT solvers. To this end, we design two novel schemes for encoding the 1-planarity of a graph (which is a problem of geometric flavor) into a discrete SAT instance. The first one is based on the Hanani-Tutte characterization of planar graphs, which has been used earlier for designing practical algorithms for upward planarity [15]. This type of encoding is rather flexible and allows to easily extend the algorithm to other use cases, such as 1-planarity of directed graphs. Our second encoding is tailored to 1-planar graphs and exploits a connection to book embeddings. It results in a specifically compact encoding and much faster processing times, and thus, can be an interesting contribution on its own.

We summarize the main contributions of the paper as follows.

- In Section 2 we describe *our algorithm for the recognition of 1-planar graphs*. It starts with a preprocessing step and an analysis of global properties of the input graph, such as edge density, that can eliminate some obviously non-1-planar instances. Then we present the two SAT encoding schemes for 1-planarity, followed by a set of more involved rules for breaking symmetries and reducing the search space.
- An *implementation of the algorithm*, `OOPS`, is open sourced in [34]. The implementation is self-contained and requires only a `C++` compiler with `C++17` support.
- Section 3 presents an *extensive evaluation* of the new approach, comparing its efficiency on a suite of benchmark graphs with alternatives. In particular, we provide the status of 1-planarity for named Wikipedia graphs (Table 2). Furthermore, we use `OOPS` to investigate two theoretical questions concerning 1-planarity regarding the smallest non-1-planar instances for several graph families (Problem 1) and the existence of a subclass of 1-planar graphs with certain properties (Problem 2).

1.2 Other Related Work

Although the family of 1-planar graphs shares some similarities with planar graphs, there is a fundamental difference. Planarity can be characterized by Wagner's theorem (forbidden minors K_5 and $K_{3,3}$) or by Kuratowski's theorem (forbidden subdivisions of K_5 and $K_{3,3}$). In contrast, every graph admits a 1-planar subdivision, making it impossible to characterize 1-planar graphs via a Kuratowski-type theorem. Testing 1-planarity is NP-complete. The first proof of this result is given by Grigoriev and Bodlaender [24], via a reduction from the

3-partition problem. An independent proof by Korzhik and Mohar [29] uses a reduction from the 3-coloring problem for planar graphs. The recognition problem remains NP-complete even for graphs with bounded bandwidth, pathwidth, or treewidth [8]; for graphs obtained from planar graphs by the addition of a single edge [14], and for graphs with a fixed rotation system [6]. However, the problem becomes fixed-parameter tractable when parameterized by the vertex-cover number, cyclomatic number, or tree-depth [8]. Polynomial-time algorithms are known only for restricted subfamilies of 1-planar graphs [5,13,25]. The extensive literature on the topic reflects a strong interest in determining the boundary between tractable and intractable cases. Nevertheless, general-purpose algorithms that are both practical and widely applicable remain lacking.

To prove that a given graph is not 1-planar, one can attempt several approaches. One method is to show that the graph has too many edges, as a 1-planar graph with n vertices has at most $4n - 8$ edges [11]; better bounds exist for subclasses such as bipartite 1-planar graphs [26, 27]. Another approach is to leverage the chromatic number: 1-planar graphs are 6-colorable [12], so a graph requiring more colors cannot be 1-planar. Alternatively, one may identify a small non-1-planar subgraph (e.g., a complete multipartite graph [16]), or show that any drawing of the graph necessarily contains too many crossings [17]. However, once a graph passes these basic filters (consider for example a random cubic graph on 30 vertices), determining whether a 1-planar drawing exists – or proving its nonexistence – remains a challenging and elusive problem. A brute-force approach is practical only for small instances, typically those with up to 16−20 vertices, depending on the level of engineering effort. For this reason, Eppstein [46], Kobourov, Liotta, and Montecchiani [28] and Binucci, Didimo, and Montecchiani [10] discuss a need for a more powerful technique to attack larger graphs.

2 Model

2.1 Preliminaries

We consider a simple undirected graph $G = (V, E)$ with n vertices and m edges. A *drawing* Γ of G maps vertices V to distinct points of the plane and edges E to Jordan curves connecting corresponding endpoints; the curves may not pass through vertices except their endpoints. Two edges *cross* if their Jordan curves intersect in a point different from the endpoints. For the ease of notation we often identify a vertex $v \in V$ and its drawing $\Gamma(v)$ as well as an edge $e \in E$ and its drawing $\Gamma(e)$. A drawing is *planar* if every edge is crossing-free, and a graph is planar if it admits a planar drawing. A drawing of a graph partitions the plane into connected regions, called *faces*; the unbounded region is called the *outer face*. The set of all faces describes the *embedding* of a planar graph G.

A graph is *1-planar* if it admits a 1-planar drawing, that is, a drawing in which every edge is crossed at most once. The *planarization* of a 1-planar drawing of G is a (planar) graph that replaces each pair of crossing edges, $(u, v) \in E$ and $(x, y) \in E$, by four edges $(u, d), (v, d), (x, d), (y, d)$, where d is a new *dummy* vertex. The vertices of G in the planarization are referred to as *original*. There are several important subclasses of 1-planar graphs; for example, *optimal* 1-planar graphs containing the maximum possible number of edges, $4n - 8$ [11]. A graph is *IC-planar* (independent crossing planar) if it has a 1-planar embedding so that each vertex is incident to at most one crossing edge [1]. In *NIC-planar* graphs (near-independent crossing planar), two pairs of crossing edges share at most one vertex [39]. *Outer 1-planar* graphs are another subclass of 1-planar graphs; they admit a 1-planar drawing such that all vertices are in the outer face [5, 25].

2.2 Preprocessing

We may simplify the testing of 1-planarity of an input graph G by preprocessing the graph and eliminating some instances early. First, we split G into (edge-disjoint) biconnected components and test each component independently, as a 1-planar drawing of G can be easily composed from 1-planar drawings of its biconnected components. Second, we check the edge density of G and reject instances violating the maximum possible density in a 1-planar graph. While there are several density-related results in the area (e.g., for IC and NIC graphs [7, 40]), we apply only the $4n - 8$ bound for general graphs [11] and the $3n - 8$ bound for bipartite ones [27]. Third, we test if G is *planar*, which can be done efficiently. Finally, we verify if G is 1-planar with *skewness* 1, that is, if it admits a 1-planar drawing in which only one pair of edges cross. This can be done by enumerating all possible pairs of edges, and testing if replacing the pair of crossed edges with a vertex of degree four yields a planar graph.

After the preprocessing step, we may assume that the input graph is biconnected, non-planar, and does not violate the edge density.

2.3 SAT-based Encoding of 1-Planarity

To test whether a given graph $G = (V, E)$ admits a 1-planar drawing, we formulate a Boolean Satisfiability Problem that has a solution if and only if G is 1-planar. We develop two encoding schemes for the problem; both are based on the concept of a linear graph layout (also referred to as an ordered embedding in [15]). Refer to [35] for a survey of various types of linear layouts. Assume that G admits a 1-planar drawing; planarize it by adding a dummy vertex for every crossing and consider a straight-line planar drawing of the planarization. Now it is easy to perturb the vertices such that they have different x-coordinates, which yield a total (linear) order of the vertices; see Figure 1. Observe that the drawing can be modified so that the vertices lie on the same horizontal line (called the *spine*) while keeping planarity and the vertex x-coordinates. The edges in the drawing are curves that are *monotone* in the x direction; they cannot cross each other but can pass through the spine multiple times, as the red edge (c, e) in Figure 1c.

Our encoding schemes operate with two auxiliary graphs built as follows. Subdivide every edge of G with a new *division* vertex to get its subdivision $G' = (V', E')$ with $|E'| = 2m$ and $|V'| = n + m$. The division vertices are $D = V' \setminus V$. From G' we build another graph, denoted G'_p, by allowing to *merge* pairs of division vertices. Let $(u, v) \in E$ and $(x, y) \in E$ be two edges of G that are subdivided by $d_{uv} \in D$ and $d_{xy} \in D$ in G'. By merging d_{uv} and d_{xy}, we get a new (dummy) vertex, d, and four new edges in G'_p, namely (u, d), (v, d), (x, d), and (y, d). The construction of G'_p is similar to planarization of (a 1-planar drawing of) G, except that G'_p contains extra non-merged division vertices. The importance of G'_p is shown below.

▶ **Observation 1.** *Let G be a graph and G' be its subdivision. Then G is 1-planar if and only if merging some pairs of division vertices in G' results in a planar graph G'_p.*

Proof. In one direction, if G is 1-planar, consider its planarization. Subdividing every edge non-incident to a dummy vertex, yields a desired planar graph G'_p. In another direction, start with a planar drawing of G'_p; *smooth* all (non-merged) degree-2 division vertices and replace all degree-4 dummy vertices with two edges. The result is a 1-planar drawing of G. ◀

To model the merging process in G' by a SAT formula, consider a partial order σ on V'. In this order all vertices are pairwise comparable, except for pairs of merged division vertices that "share" a position in σ; these exceptional pairs correspond to edge crossings in G. Formally, we introduce variables encoding the relative order of vertices V' in σ:

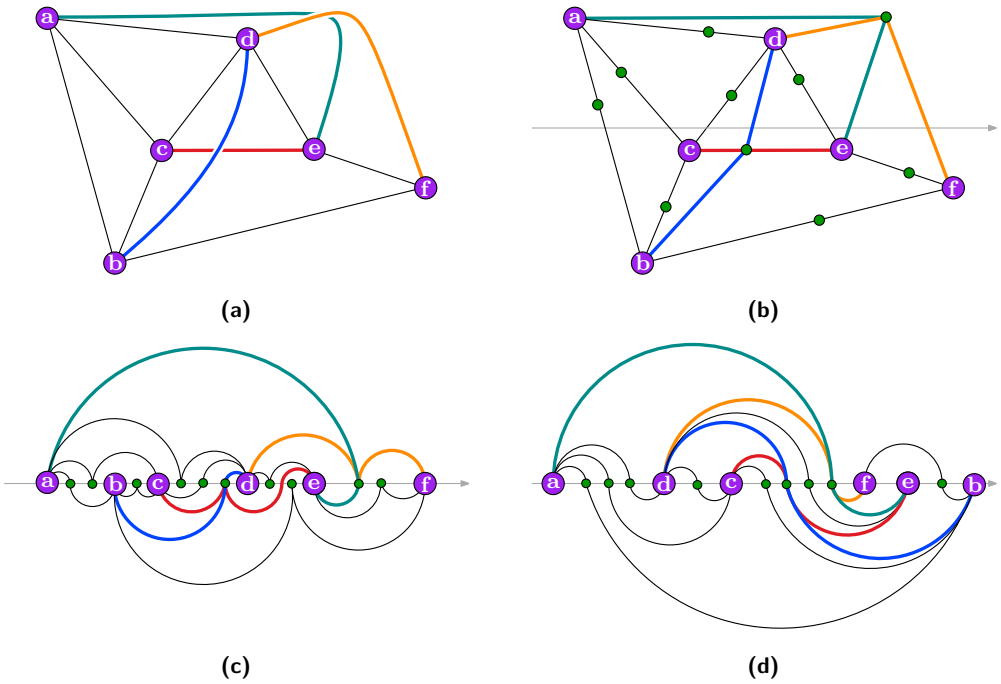


Figure 1 (a) A 1-planar drawing of a graph G with crossings between edges (a, e) and (d, f) and between (b, d) and (c, e). (b) An auxiliary graph G'_p with green *division* vertices subdividing every edge. (c) A linear layout (a.k.a. ordered embedding) of the graph for the Hanani-Tutte-based SAT encoding of 1-planarity. (d) A stack layout of the graph for the Stack-based encoding.

$\mathbb{V}1$: two variables $\sigma(v, u)$ and $\sigma(u, v)$ for every pair of distinct vertices $v, u \in V'$ indicating whether v precedes u in the order.

Intuitively, $\sigma(v, u) = \texttt{true}$ means that v precedes u in the order, and symmetrically, $\sigma(v, u) = \texttt{false}$ means that v follows u. We stress that this is a partial order, meaning that for some pairs of division vertices, called *merged*, it holds that $\sigma(v, u) = \sigma(u, v) = \texttt{false}$.

To indicate that a pair of division vertices is merged (or equivalently, that a pair of edges in G cross each other), we introduce variables for pairs of division vertices. Note that not all division vertices can be merged. For example, one may assume that adjacent edges in G do not cross in its 1-planar drawing; hence, we forbid to merge the corresponding division vertices by introducing a set of *candidate* pairs: $C \subseteq D \times D$. In the simplest scenario, C contains pairs of division vertices corresponding to independent edges in E; Section 2.4 shows how to further shrink the set of candidates. The introduced variables are as follows:

$\mathbb{V}2$: a variable $\chi(v, u)$ for every pair of distinct division vertices $(v, u) \in C$ indicating whether v is merged with u.

To ensure the correctness of the order, σ, we enforce the following constraints for the associativity for non-candidate pairs of vertices, and order transitivity:

$\mathbb{C}1$: $\sigma(v, u) \leftrightarrow \neg\sigma(u, v)$ $\quad \forall (u, v) \in V' \times V' \setminus C$;

$\mathbb{C}2$: $\neg\sigma(v, u) \vee \neg\sigma(u, v)$ $\quad \forall (u, v) \in C$;

$\mathbb{C}3$: $\big(\sigma(v, u) \wedge \sigma(u, w)\big) \rightarrow \sigma(v, w)$ $\quad \forall$ distinct $u, v, w \in V'$.

In order to link relative variables, $\mathbb{V}1$, with merge variables, $\mathbb{V}2$, we use:

$\mathbb{C}4$: $\sigma(v, u) \vee \sigma(u, v) \vee \chi(v, u)$ $\quad \forall (u, v) \in C$.

Finally, a division vertex can be merged with only one other division vertex:

$\mathbb{C}5$: $\neg\big(\chi(v, u) \wedge \chi(v, w)\big)$ $\quad \forall$ distinct $u, v, w \in D$.

Next we discuss how the basic scheme can be customized in two different ways to encode 1-planarity of a graph.

2.3.1 Hanani-Tutte Encoding

The encoding presented in the previous section allows to model a linear layout of a graph and crossings between pairs of edges. However, one still needs to guarantee that the edges can be routed in a planar way. This is achieved with a help of a Hanani-Tutte-type characterization for monotone drawings:

▶ **Lemma 2** ([23, 33]). *Let G be a graph. If G has an x-monotone drawing such that every pair of independent edges crosses an even number of times, then there is an x-monotone planar drawing of G with the same vertex locations.*

Based on the characterization, Chimani and Zeranski [15] designed a SAT formulation for *upward planarity*, where edges of a directed acyclic graph have to be drawn in a monotone fashion. We observe that essentially the same encoding can be used for graph G'_p. To this end, we need extra variables to encode (e, v)-moves that model how an edge $e \in E'$ is routed around a vertex $v \in V'$. The `true` (resp., `false`) value of the variable indicates that edge e is routed above (below) v. We refer to [23] and [15] for further details on (e, v)-moves.

$\mathbb{V}$**3:** a variable $\psi(e, v)$ for every edge $e \in E'$ and every vertex $v \in V'$ to indicate whether we perform an (e, v)-move.

The move variables can guarantee planarity using the following constraints. Here we consider a pair of edges $e = (s_e, t_e) \in E'$, $f = (s_f, t_f) \in E'$ and assume that s_e precedes t_e in σ and that s_f precedes t_f in σ.

$\mathbb{C}$**6:** $\big(\sigma(s_e, s_f) \wedge \sigma(s_f, t_e) \wedge \sigma(t_e, t_f)\big) \rightarrow \big(\psi(e, s_f) \leftrightarrow \neg\psi(f, t_e)\big)$ $\quad \forall e, f \in E'$.

$\mathbb{C}$**7:** $\big(\sigma(s_e, s_f) \wedge \sigma(s_f, t_f) \wedge \sigma(t_f, t_e)\big) \rightarrow \big(\psi(e, s_f) \leftrightarrow \psi(e, t_f)\big)$ $\quad \forall e, f \in E'$.

Finally, we need to ensure that merged division vertices have the same move variables:

$\mathbb{C}$**8:** $\chi(u, v) \rightarrow \big(\psi(u, e) \leftrightarrow \psi(v, e)\big)$ $\quad \forall (u, v) \in C, e \in E'$.

▶ **Theorem 3.** *Let $G = (V, E)$ be a graph and $G' = (V', E')$ be its subdivision. Then G is 1-planar if and only if the SAT formula built for G' and comprised of variables $\mathbb{V}1 \cup \mathbb{V}2 \cup \mathbb{V}3$ and constraints $\mathbb{C}1 \cup \mathbb{C}2 \cup \mathbb{C}3 \cup \mathbb{C}4 \cup \mathbb{C}5 \cup \mathbb{C}6 \cup \mathbb{C}7 \cup \mathbb{C}8$ is satisfiable.*

The proof of the theorem is inspired by results (for planar graphs) in [15, 23] with the key difference that G' is non-planar and contains pairs of division vertices that can be merged.

Proof of Theorem 3. Assume that graph $G = (V, E)$ is 1-planar and let $G' = (V', E')$ be its subdivision. Let us show that there is an assignment of variables $\mathbb{V}1 \cup \mathbb{V}2 \cup \mathbb{V}3$ such that $\mathbb{C}1 \cup \mathbb{C}2 \cup \mathbb{C}3 \cup \mathbb{C}4 \cup \mathbb{C}5 \cup \mathbb{C}6 \cup \mathbb{C}7 \cup \mathbb{C}8$ are satisfiable.

Consider a 1-planar drawing of G; planarize it by inserting a dummy vertex for every crossing to get a planar graph G_p. Now we subdivide every edge between original vertices and call the result G'_p. This graph, G'_p, is planar and hence, admits a straight-line drawing. Perturb the vertices in the drawing so that they have distinct x-coordinates; we get a (planar) drawing of G'_p, where every edge is monotone in the x direction; see Figure 1c. Now we build a partial order, σ, of V' by assigning $\sigma(u, v) =$ `true` whenever $x(u) < x(v)$. Dummy vertices in G'_p correspond to two division vertices in V' and have the same coordinates; set $x(u) = x(v)$ for such pairs. This construction guarantees constraints $\mathbb{C}1$, $\mathbb{C}2$, $\mathbb{C}3$, $\mathbb{C}4$, and $\mathbb{C}5$. The move variables, $\mathbb{V}3$, are assigned depending on whether an edge $e \in E'$ is routed above or below vertex $v \in V'$. It is easy to verify (see also [15, 23]) that the absence of crossings imply $\mathbb{C}6$, $\mathbb{C}7$, $\mathbb{C}8$.

For the opposite direction, assume that variables $\mathbb{V}1$, $\mathbb{V}2$, $\mathbb{V}3$ are assigned so that $\mathbb{C}1$, $\mathbb{C}2$, $\mathbb{C}3$, $\mathbb{C}4$, $\mathbb{C}5$, $\mathbb{C}6$, $\mathbb{C}7$, $\mathbb{C}8$ are satisfied for G'. Constraints $\mathbb{C}1$, $\mathbb{C}2$, and $\mathbb{C}3$ guarantee a partial order of V' where all pairs of vertices are comparable except merged pairs. We position the vertices of G' on a spine following the order. To draw edges, we use $\mathbb{V}3$ together with $\mathbb{C}6$, $\mathbb{C}7$, $\mathbb{C}8$ and apply Theorem 4.1 of [23]. Note that edges might cross the spine multiple times but do not cross each other. To get a 1-planar drawing of G, simply smooth (degree-2) division vertices; merged division vertices correspond to edge crossings. ◀

2.3.2 Stack-Based Encoding

While the Hanani-Tutte-based encoding already provides a practical approach for 1-planarity, it is not utilizing all properties of 1-planar graphs. We observe that the auxiliary graph G' is *bipartite*, with one part being the original vertices, V, and another part being the division vertices, D. Furthermore, the graph remains bipartite after merging division vertices; that is, G'_p is bipartite too. To exploit the bipartiteness, we use the concept of (2-page) *book embeddings* (a.k.a. *stack layouts*), which are special types of linear graph layouts in which edges are represented by (x-monotone) semi-arcs that do not cross each other and the spine. We rely on the following result by Felsner, Fusy, Noy, and Orden [21]:

▶ **Lemma 4** ([21]). *Let $G = (V_1 \cup V_2, E)$ be a bipartite planar graph. Then G admits a (crossing-free) stack layout with order $<$ in which edge $(u, v) \in E$ with $u < v$ (resp., $v < u$) is drawn as a semi-arc above (resp., below) the spine.*

The lemma yields a 2-page book embedding for a maximal planar bipartite graph (a quadrangulation) using its *separating decomposition*. Intuitively, a separating decomposition for a quadrangulation with a fixed planar embedding is an orientation and a (red/blue) coloring of the edges such that every vertex (except two special vertices on the outer face) is incident to a non-empty interval of red edges followed by a non-empty interval of blue edges in the cyclic order around the vertex. Such a decomposition naturally defines an *equatorial line* that separates blue and red edges. The properties of the separating decomposition imply Lemma 4, that is, the existence of a 2-page book embedding in which the equatorial line is a spine and red/blue edges form two trees, that are called *alternating* in [21]. One tree contains the edges above the spine and another one contains the edges below the spine; see Figure 1d. We refer to Figure 2 for an illustration.

In order to apply the result, assume that G is 1-planar. Then combining Observation 1 and Lemma 4, it follows that G'_p admits a stack layout with two alternating trees. This enables to encode 1-planarity of G as follows. Consider a pair of independent edges $e = (o_e, d_e) \in E'$ with $o_e \in V, d_e \in D$ and $f = (o_f, d_f) \in E'$ with $o_f \in V, d_f \in D$; assume that o_e precedes o_f in order σ. We forbid a crossing between e and f above or below the spine using the following constraints.

$\mathbb{C}9$: $\neg\sigma(o_e, o_f) \vee \neg\sigma(o_f, d_e) \vee \neg\sigma(d_e, d_f) \qquad \forall e, f \in E';$

$\mathbb{C}10$: $\neg\sigma(d_e, d_f) \vee \neg\sigma(d_f, o_e) \vee \neg\sigma(o_e, o_f) \qquad \forall e, f \in E'.$

▶ **Theorem 5.** *Let $G = (V, E)$ be a graph and $G' = (V', E')$ be its subdivision. Then G is 1-planar if and only if the SAT formula built for G' and comprised of variables $\mathbb{V}1 \cup \mathbb{V}2$ and constraints $\mathbb{C}1 \cup \mathbb{C}2 \cup \mathbb{C}3 \cup \mathbb{C}4 \cup \mathbb{C}5 \cup \mathbb{C}9 \cup \mathbb{C}10$ is satisfiable.*

Proof. Assume a graph $G = (V, E)$ is 1-planar and $G' = (V', E')$ be its subdivision. Consider a 1-planar drawing of G and build graph G'_p by planarizing the drawing and subdividing non-crossed edges. Since G'_p is bipartite, there exists a stack layout given by Lemma 4. We

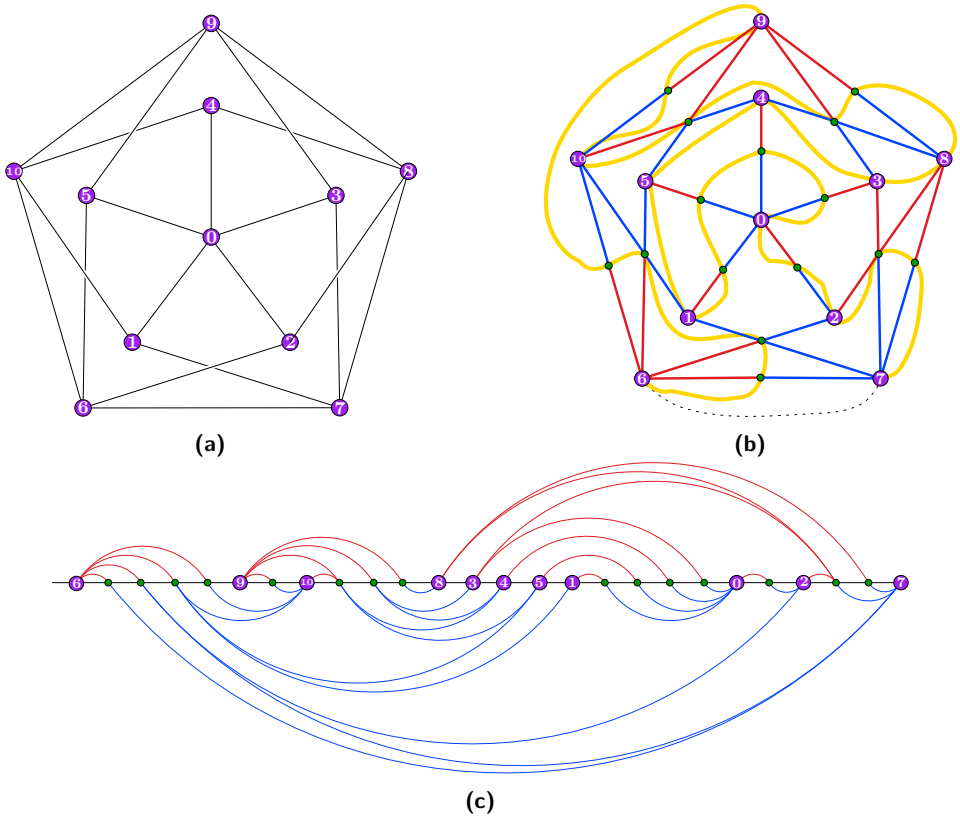


Figure 2 An illustration of the stack-based encoding for the `Grötzsch` graph: (a) A 1-planar drawing; (b) Separating decomposition with equatorial line; (c) The corresponding 2-page book embedding (stack layout) provided by Lemma 4.

use the order of vertices along the spine in the stack layout to assign variables $\mathbb{V}1$ and $\mathbb{V}2$, satisfying constraints $\mathbb{C}1$, $\mathbb{C}2$, $\mathbb{C}3$, $\mathbb{C}4$, and $\mathbb{C}5$. Since the edges do not cross each other, constraints $\mathbb{C}9$ and $\mathbb{C}10$ are also satisfied.

For the converse, note that variables $\mathbb{V}1$ and $\mathbb{V}2$ together with $\mathbb{C}1$, $\mathbb{C}2$, $\mathbb{C}3$, $\mathbb{C}4$, $\mathbb{C}5$ define a linear layout of G'_p, while $\mathbb{C}9$ and $\mathbb{C}10$ guarantee that there are no crossings between edges of G'_p. To get a 1-planar drawing of G, smooth division vertices so that every edge of G is represented by a pair of semi-arcs; merged division vertices correspond to edge crossings. ◀

2.4 Optimizations

Here we describe additional tricks to reduce the search space and speed up SAT solvers for recognizing 1-planar graphs. The first one is an improvement to relative variables, $\mathbb{V}1$. Observe that constraint $\mathbb{C}1$ enforces associativity for pairs of vertices that cannot be merged, including for example, pairs of original vertices. While modern SAT solvers are able to simplify a formula substituting variable $\sigma(v,u)$ with $\neg\sigma(u,v)$ (or vice versa), we found that performing this substitution directly during formula generation reduces both memory usage and solver runtime.

For the next optimization, we exploit a well-known property of 1-planar drawing. Consider a pair of crossing edges $e = (u,v) \in E$ and $f = (x,y) \in E$; then neither of the four adjacent edges (also called the *kite* edges in [10]) $(u,x), (x,v), (v,y), (y,u)$ can participate in a crossing. This leads to a set of extra constraints that forbid such a crossing: $\neg\chi(e,f) \vee \neg\chi(k,t)$, where $k \in E$ is one of the kite edges corresponding to pair e, f and $t \in E$ is an arbitrary edge of G.

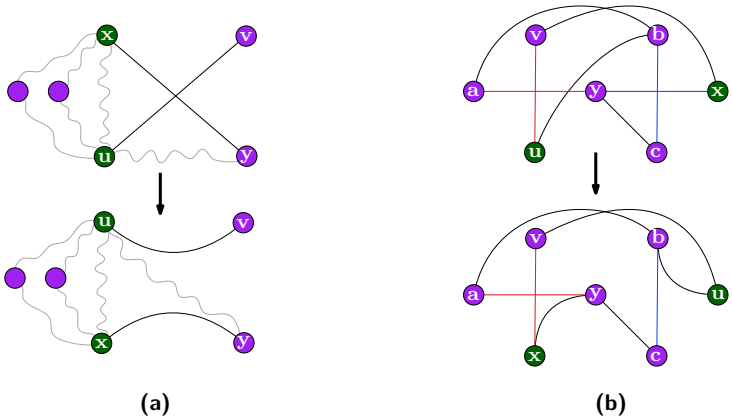


Figure 3 Possible configurations in which swapping (exchanging) x and u eliminates a crossing.

A particularly useful technique for improving the effectiveness of our approach is reducing the size of candidate pairs of edges that can cross in a 1-planar drawing, C. To this end, consider a potential crossing between edges $e = (u, v) \in E$ and $f = (x, y) \in E$. In certain cases, it might be possible to exchange the positions of u and x (*swap* u and x) while keeping the remaining vertices of G unchanged so as to eliminate the crossing. Consider the neighborhoods (set of adjacent vertices) of u and x, that is, $N(u) = \{z : (u, z) \in E\}$ and $N(x) = \{z : (x, z) \in E\}$. If $N(u) \setminus \{v, x, y\}$ coincides with $N(x) \setminus \{u, v, y\}$ and at most one edge out of $(x, v), (u, y)$ is present in E, then one can swap u and x and eliminate the crossing from the drawing; see Figure 3a. Hence, we may assume that $\chi(e, f) = \texttt{false}$ and remove the pair from the candidate set C. An extension of the rule can be applied for multiple crossing pairs. For example, if there are two crossings between $(u, v) \in E$ and $(a, y) \in E$ and between $(c, b) \in E$ with $(x, y) \in E$ and $N(u) = N(x)$ (see Figure 3b), then again swapping u and x reduces the number of crossings. In our implementation, we enumerate all possible pairs of crossings and test if swapping two vertices reduces the number of crossings; the corresponding 2-clauses forbidding one of the crossings are then added to the formula.

Finally, we observe that when a 1-planar drawing of a graph exists, it is not unique. In other words, there might be multiple satisfying assignments for a formula. To reduce the search space explored by a SAT solver, we employ *symmetry breaking* techniques; a similar approach has been used by Bekos, Kaufmann, and Zielke [9] in the context of finding stack layouts of graphs. It is easy to see that one chosen vertex, say $v_0 \in V$, can be assumed to be the first in the order. This is enforced by constraints $\sigma(v_0, u) = \texttt{true}$ for all $u \in V' \setminus v_0$. Furthermore, we may force a relative order for *twin* vertices $u, v \in V'$ whose neighborhoods are identical; that is, $\sigma(u, v) = \texttt{true}$ for pairs of twins u, v. Last but not least, we integrate into the implementation of `OOPS` two generic symmetry-breaking engines, that can analyze the entire formula and add extra constraints without changing its satisfiability.

2.5 Extensions

It is possible to extend the schemes provided in Sections 2.3.1 and 2.3.2 to several other tasks in the area of 1-planarity. Recall that a graph is NIC-planar (resp., IC-planar) if it admits a 1-planar drawing in which two pairs of crossing edges share at most one (resp., zero) vertices. This can be straightforwardly formulated using the following constraints: Let $d_i \in D$ be a division vertex adjacent to original vertices u_i and v_i. We consider distinct division vertices d_i, d_j, d_k, d_t and whenever $|\{u_i, v_i, u_j, v_j\} \cap \{u_k, v_k, u_t, v_t\}| > 1$ (resp., > 0), we introduce constraint $\neg\chi(d_i, d_j) \vee \neg\chi(d_k, d_t)$ to forbid a crossing for one of the pairs.

Another interesting extension is for *directed acyclic* graphs (DAG), whose 1-planarity has been recently studied [3, 4]. In this context, the input is a DAG and the question is whether it admits an *upward* drawing with at most one crossing per edge; a drawing is upward if for every (directed) edge (u, v), the y-coordinate of u is less than the y-coordinate of v. Our Hanani-Tutte encoding scheme can be applied for the setting by enforcing additional directional constraints $\sigma(u, v) = \texttt{true}$ for every edge (u, v) of the DAG.

3 Experiments

To validate the effectiveness of `OOPS`, we conduct an experimental evaluation focusing on three objectives. (i) Analyze how well does the algorithm perform in terms of quality and scalability compared to alternatives. (ii) Study how available parameters of the algorithm contribute to its performance. (iii) Show how `OOPS` can be used as a discovery tool to answer open research questions in graph drawing and topological graph theory.

The experiments presented in this section were conducted in two modes referred to as *single-core* and *parallel*. For the former, we utilize a Linux-based laptop with a 3.6 GHz Intel Core Ultra 5 125U processor having 16GB RAM. For the latter, we use a more powerful server with a dual-node multi-core 3.2 GHz Intel Xeon Platinum 8488C (Sapphire) processor having 360GB RAM. Single-core computations are based on the (sequential) `MapleGlucose` SAT solver [37, 41], while parallel experiments use up to 32 cores of the server machine running `Painless` SAT solver [30, 42]. To enhance SAT solving, one can optionally use integrated symmetry-breaking engines, `BreakID` [18] and `satsuma` [2].

Our algorithm, `OOPS`, is fully open sourced [34]; the implementation is self-contained (except for the optional parallel mode, which requires a parallel SAT solver) and can be built with a `C++` compiler having `C++17` support.

3.1 Comparison with Alternatives

Here we analyze how the SAT-based 1-planarity solver compares to alternative solutions. Arguably the simplest approach for testing 1-planarity is based on an exhaustive search, which enumerates all possible combinations of crossing edges, inserts a dummy vertex for every crossing, and tests whether the resulting graph is planar. We call the heuristic `brute-force`; refer to [34] for the implementation. Another candidate solver is by Binucci, Didimo, and Montecchiani [10], which we refer to as `1PlanarTester`. Since the solver is not open sourced[1], we use the same datasets and report the measurements presented in the paper [10]. The algorithms are compared with two versions of our SAT-based solver, described in Section 2.3.1 and Section 2.3.2; they are referred to as `OOPS` (H-T) and `OOPS` (Stack), respectively[2].

We use the following datasets to compare the solvers:

- NORTH50 graphs is a subset of the NORTH benchmark [45] containing all instances with at most 50 vertices; all planar graphs have been removed from the collection, resulting in a total of 297 instances;
- ROME50 graphs is a set of all non-planar ROME graphs [45] with at most 50 vertices; it contains 2950 instances.

[1] There exists an implementation of the algorithm at `https://github.com/seemanne/1PlanarTester` but a close inspection of the source code reveals substantial gaps both in performance and correctness; thus, we do not use it in our evaluation.

[2] Further heuristics are being developed independently by Fink, Münch, Pfretzschner, and Rutter [22].

Table 1 Comparison of various heuristics for 1-planarity on real-world and synthetic benchmarks. The runtimes (in seconds) are represented by mean values along with 95% confidence intervals. The best results in each category are bold.

benchmark	algorithm	1-planar		non-1-planar		unsolved
		instances	runtime, sec	instances	runtime, sec	instances
NORTH50	`1PlanarTester`	117	0.15	26	0.15	154
	`brute-force`	156	8.0 ± 12.8	28	0.1 ± 0.1	113
	`OOPS` (H-T)	**190**	3.9 ± 2.6	29	0.1 ± 0.1	78
	`OOPS` (Stack)	**190**	0.2 ± 0.1	**89**	9.5 ± 5.9	**18**
ROME50	`1PlanarTester`	412	0.15	0		2538
	`brute-force`	1961	12.5 ± 6.5	0		989
	`OOPS` (H-T)	2815	7.8 ± 0.7	0		135
	`OOPS` (Stack)	**2887**	1.3 ± 0.3	0		**63**
3REG-GIR7	`brute-force`	0		0		546
	`OOPS` (H-T)	41	105 ± 15	0		505
	`OOPS` (Stack)	**411**	75 ± 5	0		**135**
5REG-B	`brute-force`	0		0		100
	`OOPS` (H-T)	0		1	160	99
	`OOPS` (Stack)	0		**73**	90 ± 12	**27**

In addition to the real-world graphs, we selected two datasets of graphs containing 45 edges. The synthetic graph data is generated with the `geng` tool from `nauty` [31, 43].

- 3REG-GIR7 are all 546 connected cubic (3-regular) graphs with $n = 30, m = 45$ and having girth 7; all but five instances are known to be 1-planar;
- 5REG-B is a random sample of 100 connected 5-regular bipartite graphs with $n = 18, m = 45$; all the graphs are non-1-planar.

We report the results of the experiment in Table 1, which contain the number of identified 1-planar and non-1-planar instances, along with the mean processing times (in seconds) and 95% confidence intervals, for each of the benchmarks. To run the `brute-force` and `OOPS` solvers on the data, we use the *single-core* mode (which is comparable to the configuration reported in [10]) and set the runtime limit for each execution to 3 *minutes*; for comparison, the runtime limit of `1PlanarTester` in [10] is set to 3 *hours*.

One prominent observation from the results is that there is a large gap between SAT-based approaches (`OOPS`) and the naïve ones (`brute-force` and `1PlanarTester`). In most of the cases, `OOPS` is able to solve substantially more instances, both in the 1-planar and non-1-planar category. The relatively low runtime of `1PlanarTester` suggests that it can handle only "easy" instances that do not require exploring a large search space. Similarly, the performance of `brute-force` on ROME50, where it solves two thirds of the cases, implies that the collection contains many almost-planar graphs that require two or three crossings. The difference becomes striking for the "harder" collections, 3REG-GIR7 and 5REG-B, where the exhaustive search is not able to solve *any* instance, while `OOPS` (Stack) solves a majority of cases. For this reason, we do not consider existing algorithms as a viable alternative to the new SAT-based technique.

A comparison between Hanani-Tutte and Stack SAT encodings reveals a significant advantage of the latter on all the benchmarks. While both encodings contain $\Theta((n+m)^2)$ variables and $\Theta((n+m)^3)$ constraints, the Stack-based encoding is more succinct, requiring approximately half as many constraints on the data. (Note however, that Hanani-Tutte-based encoding might be more flexible, as discussed in Section 2.5). Thus, we use the Stack-based encoding as a default one in the following experiments.

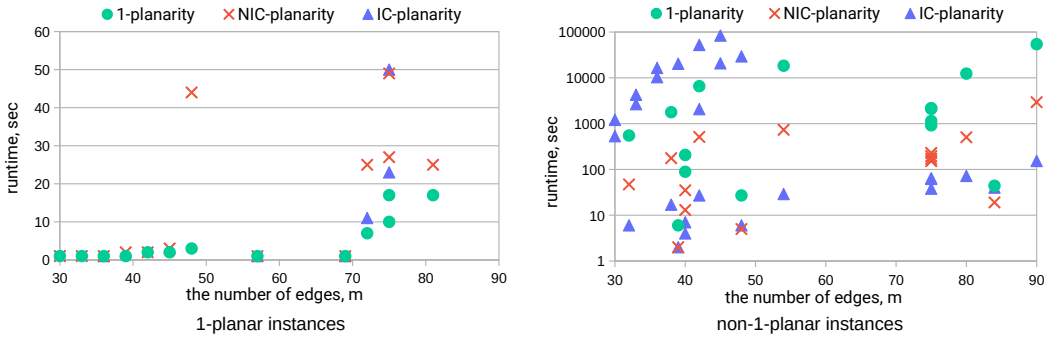


Figure 4 The runtime of OOPS (in seconds) on (a) 1-planar and (b) non-1-planar named WIKIPEDIA graphs as a function of graph size, m. Observe that in (b) the runtimes is in logarithmic scale.

3.2 Named Graphs

To further validate OOPS on a real-world benchmark, we apply the algorithm for a collection of named WIKIPEDIA graphs available at [44]. We filtered out all planar instances, and all dense instances with $m > 4n - 8$ that are obviously non-1-planar, and all instances with $m \geq 100$, which are likely too difficult for our algorithm. The remaining 47 graphs are tested with OOPS using the Stack-based encoding in the parallel mode. The status of 1-planarity, NIC-planarity, and IC-planarity (refer to Section 2.5) are presented in Table 2. Here we enforce a time limit of 24 hours for the SAT solver.

To the best of our knowledge, this is the first study of 1-planarity for many of the graphs in the dataset. OOPS is able to determine the status in the majority of the cases. Interestingly, all the unsolved instances (marked TLE in the table) correspond to cubic (3-regular) graphs. To understand the performance of OOPS on the data, we plot its runtime as a function of the number of edges in the graph; see Figure 4. We look separately at satisfiable SAT instances (that is, 1-planar graphs) and non-satisfiable ones. Observe that on the benchmark data, *every* 1-planar graph takes less than 20 seconds (and less than 50 seconds for NIC/IC-planar instances). In contrast, most of the non-1-planar instances require substantially more time: the median runtime is around 4 *minutes*, the mean is over 1.5 *hours*, while the maximum (measured on IC-planarity of `Tutte-Coxeter`) is 23 *hours*. The memory consumption of a SAT solver (`Painless` in this case) follows the same trend: it is below $4GB$ of RAM for all 1-planar instances but reaches 50−80GB on the hardest non-1-planar graphs. Given the amount of RAM required for processing non-satisfiable instances, it seems tempting to explore alternative SAT solving strategies, such as cloud-based or incremental, which we leave as a possible future work.

We have also evaluated the encoding optimizations presented in Section 2.4. While the optimizations have a moderate impact on the runtime of OOPS for 1-planar graphs, they are very helpful for large non-1-planar instances. For example, while proving non-1-planarity of the `Robertson` graph requires 2300 seconds using the basic encoding (Section 2.3.2), it needs 1750 seconds with the optimizations, a 24% speed up. For larger instances, the impact is even more pronounced: 30% (reduction from 2 hours to 85 minutes) for the `Brinkmann` graph, and 50% (from 10 to 5 hours) for `Holt`.

Table 2 Non-planar named WIKIPEDIA graphs containing at most $4n - 8$ edges and their 1-planarity status; taken from `https://en.wikipedia.org/wiki/List_of_graphs_by_edges_and_vertices`. TLE entries correspond to runtimes exceeding the limit of 24 hours.

graph name	vertices	edges	degrees	1-planar	NIC	IC
Petersen	10	15	[3...3]	*yes*	*yes*	*yes*
Franklin	12	18	[3...3]	*yes*	*yes*	*yes*
Tietze	12	18	[3...3]	*yes*	*yes*	*yes*
Grotzsch	11	20	[3...5]	*yes*	*no*	*no*
Heawood	14	21	[3...3]	*yes*	*yes*	*yes*
Chvatal	12	24	[4...4]	*yes*	*yes*	*no*
Möbius–Kantor	16	24	[3...3]	*yes*	*yes*	*yes*
Sousselier	16	27	[3...5]	*yes*	*yes*	*no*
Blanusa Snark (1st)	18	27	[3...3]	*yes*	*yes*	*yes*
Blanusa Snark (2nd)	18	27	[3...3]	*yes*	*yes*	*yes*
Pappus	18	27	[3...3]	*yes*	*yes*	*no*
Desargues	20	30	[3...3]	*yes*	*yes*	*no*
Flower snark (J5)	20	30	[3...3]	*yes*	*yes*	*no*
Hoffman	16	32	[4...4]	*no*	*no*	*no*
Loupekine snark (1st)	22	33	[3...3]	*yes*	*yes*	*no*
Loupekine snark (2nd)	22	33	[3...3]	*yes*	*yes*	*no*
McGee	24	36	[3...3]	*yes*	*yes*	*no*
Nauru	24	36	[3...3]	*yes*	*yes*	*no*
Truncated octahedral	24	36	[2...4]	*yes*	*yes*	*yes*
Robertson	19	38	[4...4]	*no*	*no*	*no*
Paley-13	13	39	[6...6]	*no*	*no*	*no*
F26A	26	39	[3...3]	*yes*	*yes*	*no*
Clebsch	16	40	[5...5]	*no*	*no*	*no*
Folkman	20	40	[4...4]	*no*	*no*	*no*
Brinkmann	21	42	[4...4]	*no*	*no*	*no*
Flower snark (J7)	28	42	[3...3]	*yes*	*yes*	*no*
Coxeter	28	42	[3...3]	*yes*	TLE	*no*
Double-Star snark	30	45	[3...3]	*yes*	*yes*	*no*
Tutte-Coxeter	30	45	[3...3]	TLE	TLE	*no*
Shrikhande	16	48	[6...6]	*no*	*no*	*no*
Dyck	32	48	[3...3]	*yes*	*yes*	*no*
Holt	27	54	[3...5]	*no*	*no*	*no*
Barnette–Bosák–Lederberg	38	57	[2...4]	*yes*	*yes*	*yes*
Tutte	46	69	[2...4]	*yes*	*yes*	*yes*
Great rhombicuboctahedral	48	72	[2...4]	*yes*	*yes*	*yes*
Foster cage	30	75	[5...5]	*no*	*no*	*no*
Meringer	30	75	[5...5]	*no*	*no*	*no*
Robertson–Wegner	30	75	[5...5]	*no*	*no*	*no*
Wong	30	75	[5...5]	*no*	*no*	*no*
Szekeres snark	50	75	[3...3]	*yes*	*yes*	*yes*
Watkins snark	50	75	[3...3]	*yes*	*yes*	*yes*
Wells	32	80	[5...5]	*no*	*no*	*no*
Ellingham–Horton (54)	54	81	[3...3]	*yes*	*yes*	TLE
Gray	54	81	[3...3]	TLE	TLE	TLE
Klein (7-regular)	24	84	[7...7]	*no*	*no*	*no*
Klein (cubic)	56	84	[3...3]	TLE	TLE	TLE
Sylvester	36	90	[5...5]	*no*	*no*	*no*

3.3 Applications

Arguably among the most natural applications of a tool for recognizing 1-planar graphs is finding the smallest non-1-planar instance in a family of graphs.

▶ **Problem 1.** *What is the smallest non-1-planar instance in a family of graphs?*

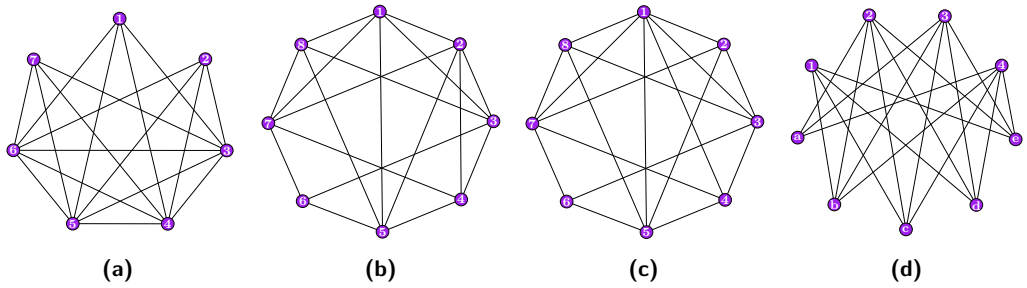


Figure 5 (a-c) The three connected non-1-planar graphs with $m = 18$ edges. (d) The unique connected bipartite non-1-planar graph with $m = 19$; it is obtained from $K_{4,5}$ by removing an edge.

This question has two flavors depending on whether one is interested in vertex-minimal or edge-minimal instances. For planar graphs, it is well-known that smallest non-planar graphs contain 5 vertices (K_5) or 9 edges ($K_{3,3}$). For 1-planar graphs, it is known that K_7 is the smallest complete non-1-planar graphs (see e.g., [16]), however, no such bound is known for non-1-planar edge-minimal graph. To fill the gap, we computationally tested all simple connected instances with up to $m = 18$ edges (generated with [43]). All graphs with up to $m = 17$ are 1-planar and there are exactly 3 (out of 164,551,477 connected) non-1-planar instances with $m = 18$; see Figure 5. For bipartite graphs, a graph on 9 vertices (in fact, $K_{4,5}$ as shown by Czap and Hudák [16]) is one of the two vertex-minimal bipartite non-1-planar graphs. The second such graph, namely $K_{4,5} \setminus e$, is in addition the unique edge-minimal bipartite non-1-planar graph; see Figure 5d. This is confirmed by testing all 25,124,511 connected bipartite graphs with $m = 19$ and all smaller instances.

A related question (discussed in [47]) is: *What is the minimum difference between edge and vertex counts in a connected non-1-planar graph?* For planar graphs, one can show (see [47]) that every connected graph with $m \leq n + 2$ is planar, while non-planar $K_{3,3}$ has $m = n + 3$. For 1-planarity, such a condition is unknown. Existing non-1-planar graphs, such as $K_{4,5}$ and $K_{3,7}$ [16], indicate that the difference can be $m - n = 11$. The two examples we found (Figures 5a and 5d) reduce the bound to 10. It is intriguing to fully resolve the question.

A particularly challenging variant of Problem 1 is to find the smallest 3-regular (cubic) graph that is not 1-planar. While it is easy to show (see e.g., [19]) that random (large) cubic graphs are not 1-planar, the smallest such instance is unknown.

▶ **Problem 1b** ([46, 48]). *What is the smallest cubic non-1-planar graph?*

In an attempt to answer the question, Eppstein [46] manually constructed 1-planar drawings of several graphs and suggested that either the `Coxeter` graph or the `Tutte-Coxeter` graph is not 1-planar. As pointed out in Table 2, the former is a 1-planar instance; see Figure 6. The latter is very likely non-1-planar, but we were unable to fully verify it, despite running several (parallel) SAT solvers for days. We did, however, verify 1-planarity of (i) all cubic graphs with up to $n = 24$ vertices and of (ii) all cubic bipartite graphs with up to $n = 30$ vertices. Note that there are around 150×10^6 such instances and the total computation took $\approx$1.5 machine-months (an equivalent of $\approx$20 ms per instance).

As another application of `OOPS`, we investigate the following question proposed by Zhang, Ouyang, and Huang [38] regarding *claw-free* 1-planar graphs; recall that a claw is graph $K_{1,3}$.

▶ **Problem 2** ([38]). *Does there exist infinitely many 6-connected claw-free 1-planar graphs?*

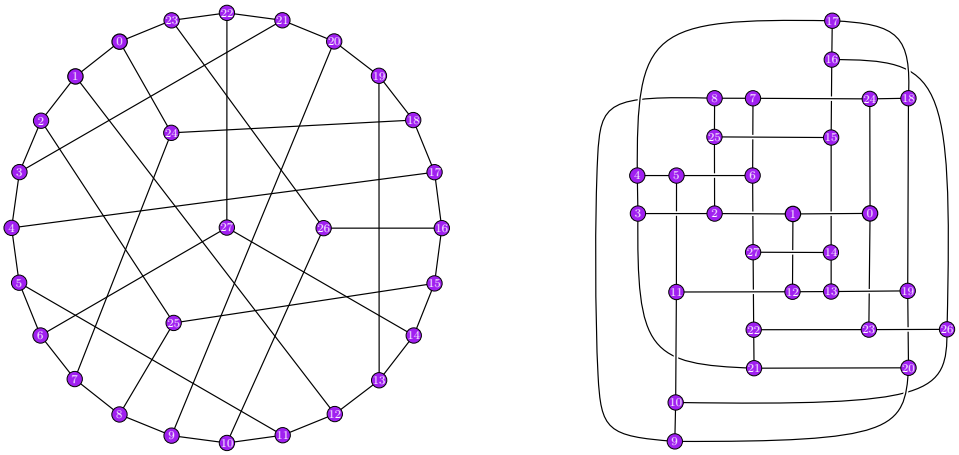


Figure 6 The `Coxeter` graph with $n = 28$ vertices and $m = 42$ graph (left) and its 1-planar drawing (right) constructed by `OOPS`.

In fact, the original version of the paper [38] suggests a candidate family, namely the cube of a cycle C_n. That is, a graph with vertex set $\{v_0, v_1, \ldots, v_{n-1}\}$ in which an edge (v_i, v_j) exists if $\min(|i-j|, n-|i-j|) \leq 3$. Such graphs are (i) claw-free, since the neighborhood of any vertex is an interval on the cycle, so no vertex can have three pairwise non-adjacent neighbors, and (ii) 6-connected (for $n \geq 7$), since to disconnect two vertices, u and v, one needs to remove three consecutive vertices going clockwise from u to v and the same counter-clockwise. However, the status of 1-planarity of such graphs is open. Using `OOPS`, we verified that such graphs are indeed 1-planar whenever $n = 8k$ for every $k > 0$.

▶ **Theorem 6.** *Let C_n^3 be a graph obtained from a cycle C_n by connecting every pair of vertices that are at most 3 edges apart in C_n. Then C_n^3 is 1-planar if $n = 8k$ for every $k > 0$.*

Proof. Let the vertices of C_n^3 for $n = 8k$ be $\{v_0, v_1, \ldots, v_{n-1}\}$. Split the vertices into $2k$ (consecutive) groups of size four, that is, $\{v_0, v_1, v_2, v_3\}, \ldots, \{v_{n-4}, v_{n-3}, v_{n-2}, v_{n-1}\}$; observe that all the edges of C_n^3 are either within the same group or between vertices of two consecutive groups (modulo $2k$). A 1-planar drawing is built by placing the vertices on a horizontal line (spine), following the order given by the groups. Within i-th group, $\{v_x, v_{x+1}, v_{x+2}, v_{x+3}\}$, place the four vertices in the order $v_x, v_{x+2}, v_{x+1}, v_{x+3}$ if i is even, and in the order $v_{x+1}, v_{x+2}, v_x, v_{x+3}$ if i is odd; see Figure 7. Because of the symmetry, we only need to specify the drawing of the edges between groups i and $i+1$, and between groups $i+1$ and $i+2$. Refer to Figure 7c for the construction; note that most edges are represented as semi-arcs (either above or below the spine) except for one inter-group edge for every pair of groups, which is a bi-arc, e.g., edges $(2, 4)$, $(6, 9)$, and $(10, 12)$ in the figure. ◀

According to `OOPS` (for small values of n), graph C_n^3 is 1-planar only if $n = 8k, k > 0$ but proving that direction might be challenging.

4 Conclusion

We presented `OOPS`, a new SAT-based algorithm for testing 1-planarity and provided its implementation [34] to facilitate further research in the field. While the algorithm is shown to be useful for resolving various problems concerning 1-planarity of specific graph families, many questions remain unanswered. One particularly challenging task, discussed in Section 3.3, is

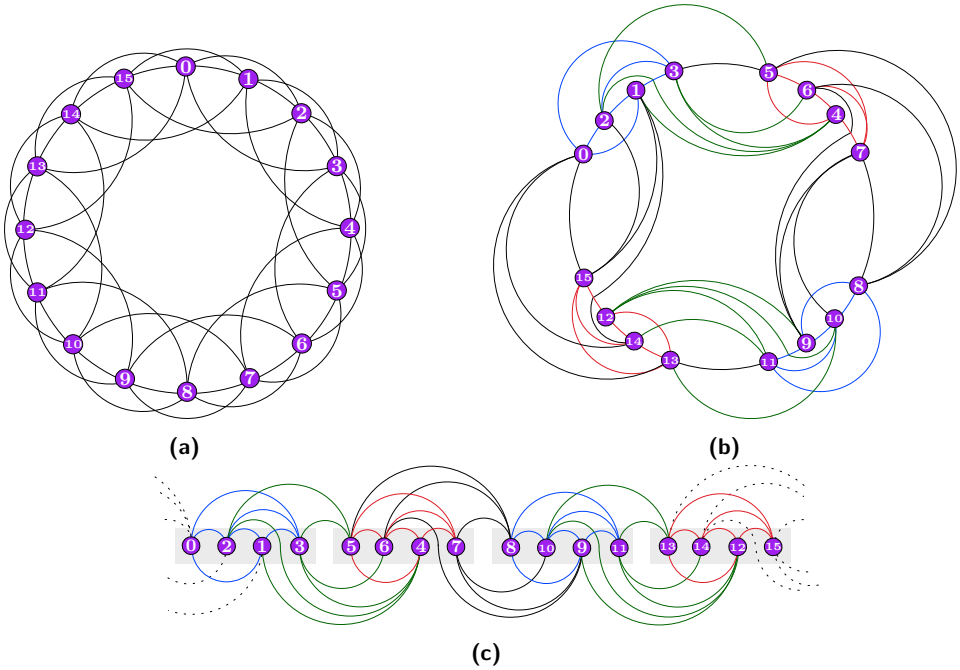


Figure 7 An illustration for Theorem 6: (a) C_{16}^3 and (b) its 1-planar drawing. (c) The edge pattern is generalized for arbitrary $n = 8k, k > 0$.

finding the smallest cubic graph that is not 1-planar. We conjecture that the `Tutte-Coxeter` graph is such an instance, along with 4 other 30-vertex graphs of girth 7, for which none of the tested (parallel) SAT solvers was able to find a solution even after a week of processing. We stress that proving the minimality would also require finding a 1-planar drawing for more than *forty billion* 28-vertex cubic graphs.

Another intriguing direction to explore is 1-planarity of DAGs; see Section 2.5 for an adaptation of `OOPS` for the case. Among many possible questions to study (refer for example to [3, 32]), we highlight one discussed by Angelini et al. [3, 4]:

▶ **Problem 3** ([3, 4]). *Is there a directed acyclic outerpath that is not upward 1-planar?*

Finally, extending our formulations or designing new ones for other classes of beyond-planar graphs, such as k-planar graphs for $k > 1$, is of interest. While there is a straightforward adaptation of our scheme to 2-planarity (by inserting more division vertices per edge and enforcing appropriate merging rules), it might not be effective in practice due to the explosion of the search space. Hence, designing further techniques for simplifying or restricting the resulting SAT formulas, in addition to the ones discussed in Section 2.4, is an avenue for future research.

References

1 Michael O Albertson. Chromatic number, independence ratio, and crossing number. *Ars Math. Contemp.*, 1(1):1–6, 2008. `doi:10.26493/1855-3974.10.2D0`.

2 Markus Anders, Sofia Brenner, and Gaurav Rattan. Satsuma: Structure-based symmetry breaking in SAT. In Supratik Chakraborty and Jie-Hong Roland Jiang, editors, *Theory and Applications of Satisfiability Testing*, volume 305 of *LIPIcs*, pages 4:1–4:23, 2024. `doi:10.4230/LIPICS.SAT.2024.4`.

3 Patrizio Angelini, Therese Biedl, Markus Chimani, Sabine Cornelsen, Giordano Da Lozzo, Seok-Hee Hong, Giuseppe Liotta, Maurizio Patrignani, Sergey Pupyrev, Ignaz Rutter, and Alexander Wolff. The price of upwardness. In *International Symposium on Graph Drawing and Network Visualization (GD)*, volume 320, pages 13:1–13:20, 2024. doi:10.4230/LIPIcs.GD.2024.13.

4 Patrizio Angelini, Therese Biedl, Markus Chimani, Sabine Cornelsen, Giordano Da Lozzo, Seok-Hee Hong, Giuseppe Liotta, Maurizio Patrignani, Sergey Pupyrev, and Ignaz Rutter. The price of upwardness. *Discret. Math. Theor. Comput. Sci.*, 27(3), 2025. doi:10.46298/DMTCS.15222.

5 Christopher Auer, Christian Bachmaier, Franz J Brandenburg, Andreas Gleißner, Kathrin Hanauer, Daniel Neuwirth, and Josef Reislhuber. Outer 1-planar graphs. *Algorithmica*, 74(4):1293–1320, 2016. doi:10.1007/S00453-015-0002-1.

6 Christopher Auer, Franz Brandenburg, Andreas Gleißner, and Josef Reislhuber. 1-planarity of graphs with a rotation system. *Journal of Graph Algorithms and Applications*, 19(1):67–86, 2015. doi:10.7155/JGAA.00347.

7 Christian Bachmaier, Franz J Brandenburg, Kathrin Hanauer, Daniel Neuwirth, and Josef Reislhuber. NIC-planar graphs. *Discret. Appl. Math.*, 232:23–40, 2017. doi:10.1016/J.DAM.2017.08.015.

8 Michael J. Bannister, Sergio Cabello, and David Eppstein. Parameterized complexity of 1-planarity. *J. Graph Algorithms Appl.*, 22(1):23–49, 2018. doi:10.7155/JGAA.00457.

9 Michael A. Bekos, Michael Kaufmann, and Christian Zielke. The book embedding problem from a SAT-solving perspective. In Emilio Di Giacomo and Anna Lubiw, editors, *Graph Drawing and Network Visualization GD*, volume 9411 of *Lecture Notes in Computer Science*, pages 125–138. Springer, 2015. doi:10.1007/978-3-319-27261-0_11.

10 Carla Binucci, Walter Didimo, and Fabrizio Montecchiani. 1-planarity testing and embedding: An experimental study. *Comput. Geom.*, 108:101900, 2023. doi:10.1016/J.COMGEO.2022.101900.

11 Rainer Bodendiek, H Schumacher, and Klaus Wagner. Über 1-optimale graphen. *Mathematische Nachrichten*, 117(1):323–339, 1984. doi:10.1002/mana.3211170125.

12 O. V. Borodin. Solution of the Ringel problem on vertex-face coloring of planar graphs and coloring of 1-planar graphs. *Metody Diskret. Analiz.*, 41(108):12–26, 1984.

13 Franz J. Brandenburg. Characterizing and recognizing 4-map graphs. *Algorithmica*, 81(5):1818–1843, 2019. doi:10.1007/S00453-018-0510-X.

14 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.

15 Markus Chimani and Robert Zeranski. Upward planarity testing in practice: SAT formulations and comparative study. *ACM J. Exp. Algorithmics*, 20:1–27, 2015. doi:10.1145/2699875.

16 Július Czap and Dávid Hudák. 1-planarity of complete multipartite graphs. *Discret. Appl. Math.*, 160(4–5):505–512, 2012. doi:10.1016/J.DAM.2011.11.014.

17 Július Czap and Dávid Hudák. On drawings and decompositions of 1-planar graphs. *Electron. J. Comb.*, 20(2):54, 2013. doi:10.37236/2392.

18 Jo Devriendt and Bart Bogaerts. BreakID: Static symmetry breaking for ASP (system description). *arXiv preprint arXiv:1608.08447*, 2016. arXiv:1608.08447.

19 Vida Dujmovic, Ken-ichi Kawarabayashi, Bojan Mohar, and David R. Wood. Improved upper bounds on the crossing number. In *Proceedings of the Twenty-Fourth Annual Symposium on Computational Geometry*, pages 375–384. ACM, 2008. doi:10.1145/1377676.1377739.

20 Eduard Eiben, Robert Ganian, Thekla Hamm, Fabian Klute, and Martin Nöllenburg. Extending partial 1-planar drawings. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *International Colloquium on Automata, Languages, and Programming*, LIPIcs, pages 43:1–43:19, 2020. doi:10.4230/LIPIcs.ICALP.2020.43.

21 Stefan Felsner, Éric Fusy, Marc Noy, and David Orden. Bijections for Baxter families and related objects. *Journal of Combinatorial Theory, Series A*, 118(3):993–1020, 2011. doi:10.1016/j.jcta.2010.03.017.

22 Simon D. Fink, Miriam Münch, Matthias Pfretzschner, and Ignaz Rutter. Heuristics for exact 1-planarity testing. In Vida Dujmović and Fabrizio Montecchiani, editors, *33rd International Symposium on Graph Drawing and Network Visualization (GD 2025)*, Leibniz International Proceedings in Informatics (LIPIcs), pages 4:1–4:19, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.GD.2025.4.

23 Radoslav Fulek, Michael J. Pelsmajer, Marcus Schaefer, and Daniel Štefankovič. Hanani-Tutte, monotone drawings, and level-planarity. In János Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 263–287. Springer, 2013. doi:10.1007/978-1-4614-0110-0_14.

24 Alexander Grigoriev and Hans L Bodlaender. Algorithms for graphs embeddable with few crossings per edge. *Algorithmica*, 49(1):1–11, 2007. doi:10.1007/s00453-007-0010-x.

25 Seok-Hee Hong, Peter Eades, Naoki Katoh, Giuseppe Liotta, Pascal Schweitzer, and Yusuke Suzuki. A linear-time algorithm for testing outer-1-planarity. *Algorithmica*, 72:1033–1054, 2015. doi:10.1007/s00453-014-9890-8.

26 Yuanqiu Huang, Zhangdong Ouyang, and Fengming Dong. On the sizes of bipartite 1-planar graphs. *Electron. J. Comb.*, 28(2):2, 2021. doi:10.37236/10012.

27 Dmitri V Karpov. An upper bound on the number of edges in an almost planar bipartite graph. *Journal of Mathematical Sciences*, 196:737–746, 2014. doi:10.1007/s10958-014-1690-9.

28 Stephen G Kobourov, Giuseppe Liotta, and Fabrizio Montecchiani. An annotated bibliography on 1-planarity. *Computer Science Review*, 25:49–67, 2017. doi:10.1016/j.cosrev.2017.06.002.

29 Vladimir P Korzhik and Bojan Mohar. Minimal obstructions for 1-immersions and hardness of 1-planarity testing. *Journal of Graph Theory*, 72(1):30–71, 2013. doi:10.1002/jgt.21630.

30 Ludovic Le Frioux, Souheib Baarir, Julien Sopena, and Fabrice Kordon. PaInleSS: A framework for parallel SAT solving. In *Theory and Applications of Satisfiability Testing*, pages 233–250. Springer, 2017. doi:10.1007/978-3-319-66263-3_15.

31 Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, ii. *Journal of Symbolic Computation*, 60:94–112, 2014. doi:10.1016/j.jsc.2013.09.003.

32 Martin Nöllenburg and Sergey Pupyrev. On families of planar dags with constant stack number. In Michael A. Bekos and Markus Chimani, editors, *Graph Drawing and Network Visualization (GD)*, pages 135–151. Springer, 2023. doi:10.1007/978-3-031-49272-3_10.

33 János Pach and Géza Tóth. Monotone drawings of planar graphs. *Journal of Graph Theory*, 46(1):39–47, 2004. doi:10.1002/jgt.10168.

34 Sergey Pupyrev. spupyrev/oops. Software (visited on 2025-11-07). URL: https://github.com/spupyrev/oops, doi:10.4230/artifacts.25056.

35 Sergey Pupyrev. A collection of existing results on stack and queue numbers. https://spupyrev.github.io/linearlayouts.html, 2020. Accessed: 5/23/2025.

36 Gerhard Ringel. Ein Sechsfarbenproblem auf der Kugel. In *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, volume 29, pages 107–117. Springer, 1965. doi:10.1007/BF02996313.

37 Vincent Vallade, Ludovic Le Frioux, Razvan Oanea, Souheib Baarir, Julien Sopena, Fabrice Kordon, Saeed Nejati, and Vijay Ganesh. New concurrent and distributed painless solvers: p-mcomsps, pmcomsps-com, p-mcomsps-mpi, and p-mcomsps-com-mpi. *SAT Competition*, page 40, 2021.

38 Licheng Zhang, Zhangdong Ouyang, and Yuanqiu Huang. Maximum degree and connectivity on claw-free 1-planar graphs, 2025. v1 accessed: 1/25/2025. arXiv:2501.15124.

39 Xin Zhang. Drawing complete multipartite graphs on the plane with restrictions on crossings. *Acta Mathematica Sinica, English Series*, 30(12):2045–2053, 2014. doi:10.1007/s10114-014-3763-6.

40 Xin Zhang and Guizhen Liu. The structure of plane graphs with independent crossings and its applications to coloring problems. *Central European Journal of Mathematics*, 11:308–321, 2013. doi:10.2478/s11533-012-0094-7.

41 MapleGlucose SAT solver. https://maplesat.github.io/solvers.html, 2021.

42 Painless SAT solver. https://www.lrde.epita.fr/wiki/Painless, 2021.
43 nauty and Traces. https://pallini.di.uniroma1.it. Accessed: 5/23/2025.
44 List of graphs by edges and vertices. https://en.wikipedia.org/wiki/List_of_graphs_by_edges_and_vertices. Accessed: 5/23/2025.
45 AT&T and Rome graphs. http://graphdrawing.org/data.html. Accessed: 5/23/2025.
46 David Eppstein. Cubic 1-planarity. https://11011110.github.io/blog/2014/05/11/cubic-1-planarity.html. Accessed: 5/23/2025.
47 Mathematics StackExchange. Lower bound on the number of edges in a non-k-planar graph. https://math.stackexchange.com/questions/4637623. Accessed: 5/23/2025.
48 Mathematics StackExchange. Can a 3-regular non-1-planar graph be constructed? https://mathoverflow.net/questions/441370. Accessed: 5/23/2025.

Crossing Number of Simple 3-Plane Drawings

Miriam Goetze
Karlsruhe Institute of Technology, Germany

Michael Hoffmann
Department of Computer Science, ETH Zürich, Switzerland

Ignaz Rutter
University of Passau, Germany

Torsten Ueckerdt
Karlsruhe Institute of Technology, Germany

Abstract

We study 3-plane drawings, that is, drawings of graphs in which every edge has at most three crossings. We show how the recently developed Density Formula for topological drawings of graphs [9] can be used to count the crossings in terms of the number n of vertices. As a main result, we show that every 3-plane drawing has at most $5.5(n-2)$ crossings, which is tight. In particular, it follows that every 3-planar graph on n vertices has crossing number at most $5.5n$, which improves upon a recent bound [3] of $6.6n$. To apply the Density Formula, we carefully analyze the interplay between certain configurations of cells in a 3-plane drawing. As a by-product, we also obtain an alternative proof for the known statement that every 3-planar graph has at most $5.5(n-2)$ edges.

2012 ACM Subject Classification Mathematics of computing → Graphs and surfaces

Keywords and phrases beyond planar graphs, edge density, crossing number, density formula

Digital Object Identifier 10.4230/LIPIcs.GD.2025.15

Funding *Miriam Goetze*: funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 520723789.
Ignaz Rutter: funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 541433306.
Torsten Ueckerdt: funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 520708409.

Acknowledgements This research was initiated at the Workshop on *Graph and Network Visualization* (GNV 2024) in Heiligkreuztal, Germany, June 23–28, 2024.

1 Introduction

One of the most basic combinatorial questions one can ask for a class of graphs is: How many edges can a graph from this class have as a function of the number n of vertices? Prominent examples include upper bounds of $\binom{n}{2}$ for the class of all graphs and $\frac{n^2}{4}$ for bipartite graphs. These bounds are immediate consequences of the definition of these graph classes, and they are tight, that is, there exist graphs in the class with exactly this many edges. But for several other graph classes good upper bounds on the number of edges are much more challenging to obtain. Notably this holds for classes that relate to the existence of certain geometric representations. One the most fundamental questions one can ask about a class of geometrically represented graphs is: What is the minimum number of edge crossings required in such a representation, as a function of the number n of vertices? We study both of these fundamental questions in combination, for the class of 3-planar graphs. A graph is *k-planar* if it can be drawn in the plane such that every edge has at most k crossings. The study of k-planar graphs goes back to Ringel [16] and has been a major focus in graph drawing over the past two decades [8], as a natural generalization of planar graphs ($k = 0$).


© Miriam Goetze, Michael Hoffmann, Ignaz Rutter, and Torsten Ueckerdt;
licensed under Creative Commons License CC-BY 4.0



The maximum number of edges in a simple k-planar graph on n vertices is known to be at most $c_k(n-2)$, where $c_0 = 3$, $c_1 = 4$ [5], $c_2 = 5$ [14, 15], $c_3 = 5.5$ [10, 11], $c_4 = 6$ [1], and $c_k \leq 3.81\sqrt{k}$, for general $k \geq 5$ [1]. The bounds for $k \leq 2$ are tight and those for $k \leq 4$ are tight up to an additive constant [1, 4]. The bounds for $k \leq 4$ also generalize to *non-homotopic* drawings of multigraphs [12, 13], that is, where every continuous transformation that transforms one copy of an edge to another passes over a vertex. Interestingly, the upper bound for 3-planar graphs is tight in this more general setting only [4, 6].

The *crossing number* of a drawing Γ is the number of edge crossings in Γ. The *crossing number* $\mathrm{cr}(G)$ of a graph G is the minimum crossing number over all drawings of G. By definition every k-planar graph G admits a k-plane drawing and thus

$$\mathrm{cr}(G) \leq \frac{km}{2}, \tag{S}$$

where m denotes the number of edges in G. For a k-planar graph, this simple inequality connects upper bounds on the number of edges with lower bounds on the crossing number. Both of these come together in the well-known Crossing Lemma [2, Chapter 45], as the best constants in the Crossing Lemma are obtained by analyzing k-plane drawings [1, 6, 10, 11]. Conversely, combining the lower bound on $\mathrm{cr}(G)$ from the Crossing Lemma with an upper bound on $\mathrm{cr}(G)$ we obtain an upper bound on the number of edges in G. While (S) would work here, it is probably not an ideal choice because the graphs for which (S) is tight might be very different from those graphs that have a maximum number of edges, for any fixed n. For instance, for a 1-planar graph G we have $\mathrm{cr}(G) \leq n-2$ [17, Proposition 4.4], which beats the bound we get by plugging $m \leq 4n-8$ into (S) by a factor of two. Can we obtain similar improvements by bounding $\mathrm{cr}(G)$ in terms of n, rather than m, for $k \geq 2$?

Indeed, very recently it has been shown that $\mathrm{cr}(G) \leq \mathrm{cr}_2(G) \leq 3.\overline{3}n$ if G is 2-planar [3, Theorem 3] and $\mathrm{cr}(G) \leq \mathrm{cr}_3(G) \leq 6.6n$ if G is 3-planar [3, Theorem 4]. (The *k-planar crossing number* $\mathrm{cr}_k(G)$ is similar to the crossing number, except that the minimum is taken over all k-plane drawings of G.) There is some indication that the bound for 2-planar graphs could be tight up to an additive constant, as it is achieved by the standard drawings of optimal 2-planar graphs (Figure 1). But the crossing number of these graphs is not known.

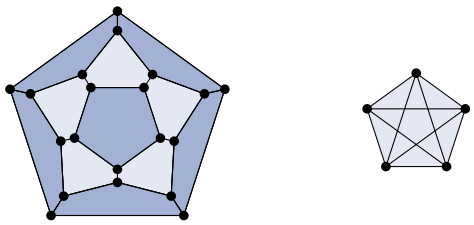


Figure 1 Construction by Pach and Tóth [15, Figure 3]. Left: A plane drawing with pentagonal faces. Right: To each pentagonal face all diagonals are added.

In contrast, there exists a family of simple 3-planar graphs with $5.5n - 15$ edges whose standard drawings have $5.5n - 21$ crossings (Figure 2). Thus, there is a gap of $1.1n$ between the lower and the upper bound for the crossing number of 3-plane drawings.

Results. We close the gap and present an upper bound on the crossing number of 3-plane drawings that is tight up to an additive constant. Using the same approach we also obtain an alternative proof to show that a 3-planar n-vertex graph has at most $5.5(n-2)$ edges.

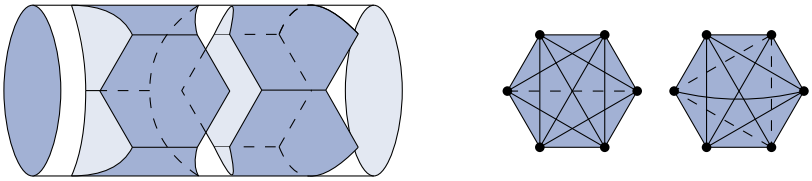


Figure 2 Construction from [11, Figure 8]. Left: A cylinder with two layers, each consisting of three hexagonal faces. Right: To each face of a layer all but one diagonal is added. To the top and bottom face six diagonals can be added without creating parallel edges. Missing diagonals are represented by dashed lines.

▶ **Theorem 1.** *Every non-homotopic 3-plane drawing of a graph on n vertices, $n \geq 3$, contains at most $5.5(n-2)$ edges and at most $5.5(n-2)$ crossings.*

Our proof relies on the recently developed Density Formula (cf. Theorem 4 below) for topological drawings of graphs [9]. It relates the number of vertices, edges, and cells of various sizes in a drawing, in a way similar to the Euler Formula in the case of plane graphs. Previously, the Density Formula has been used to derive upper bounds on the number of edges in k-plane drawings, for $k \leq 2$ [9]. In order to apply it to 3-plane drawings, to bound the number of crossings, and to obtain tight bounds, we study cells not only in isolation but also as part of what we call *configurations*, which consist of several connected cells. We then develop a number of new constraints that relate the number of cells and/or configurations of a certain type in any 3-plane drawing. The combination of all these constraints with the Density Formula yields a linear program that we can solve in two different ways – maximizing either the number of edges or the number of crossings – to prove Theorem 1.

As with other proofs involving the Density Formula [9], our technique can be seen as a variant of the discharging method. But we streamline the arguments so that we prove each constraint ad hoc by rather simple means, usually requiring only an easy double counting.

Using Theorem 1 we can derive better upper bounds on the number of edges in k-planar graphs without short cycles. Plugging our bound of at most $5.5n$ crossings into the proofs from [3] we obtain the following.

▶ **Corollary 2.** *For every integer n, each of the following holds.*

- *Every C_3-free 3-planar graph on n vertices has at most $\sqrt[3]{891/8}\,n < 4.812n$ edges (down from $\approx 5.113n$ [3, Theorem 18]).*
- *Every C_4-free 3-planar graph on n vertices has at most $\sqrt[3]{1\,254\,825/12\,544}\,n < 4.643n$ edges (down from $\approx 4.933n$ [3, Theorem 20]).*
- *Every 3-planar graph of girth 5 on n vertices has at most $\sqrt[3]{122\,793/1600}\,n < 4.25n$ edges (down from $\approx 4.516n$ [3, Theorem 21]).*

2 Preliminaries

We consider *drawings* of graphs on the sphere with vertices as points, edges as Jordan arcs, and the usual assumption that any two edges share only finitely many points, each being a common endpoint or a proper crossing, and that no three edges cross in the same point. As is customary, we do not distinguish between the points and curves in Γ and the vertices and edges of G they represent, respectively. We also assume the drawings to be *simple*[1], that is,

[1] Our definition is slightly more general than what is usually meant in the literature, namely that "any two edges have at most one point in common". Our simple drawings allow non-crossing parallel edges.

no edge crosses itself, no two adjacent edges cross, and any two edges cross at most once. We explicitly allow our graphs to contain parallel edges, but no loops. Hence, our graphs are not necessarily simple, while our drawings are simple. In order to avoid an arbitrary number of parallel edges within a small corridor, a drawing Γ is called *non-homotopic* if every region that is bounded by two parallel edges, called a *lens*, contains a crossing or a vertex in its interior; see Figure 3.

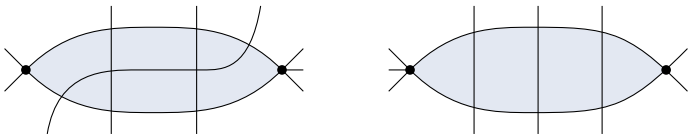


Figure 3 Left: A lens (blue) with two crossings in its interior and two vertices on its boundary. Such a lens can be part of a non-homotopic drawing. Right: An *empty* lens (blue), i.e., a lens that contains neither a vertex nor a crossing. Such a lens cannot be part of a non-homotopic drawing.

Let Γ be a drawing of a graph $G = (V, E)$. If every edge is crossed at most three times, we say that Γ is 3-*plane*. We denote the set of crossings by X. For $i \in \{0, 1, 2, 3\}$, let $E_i \subseteq E$ be the set of all edges with exactly i crossings, and let $E_\times = E_1 \cup E_2 \cup E_3$.

We easily observe the following.

▶ **Observation 3.** *If Γ is a 3-plane drawing, we have*

$$|E| = |E_\times| + |E_0| \tag{9.A}$$
$$|E_\times| = |E_1| + |E_2| + |E_3| \tag{8.A}$$
$$2|X| = |E_1| + 2|E_2| + 3|E_3| \tag{8.B}$$

Edge-Segments and Cells. An edge with i crossings is split into $i + 1$ parts, called *edge-segments*. An edge-segment is *inner* if both its endpoints are crossings, *uncrossed* if both its endpoints are vertices and *outer*, otherwise. The *planarization* of Γ is the graph obtained by replacing every crossing x with a vertex of degree 4 that is incident to the four edge-segments of x. We say that the drawing Γ is *connected*, if its planarization is a connected graph, and shall henceforth only consider connected drawings. Removing all edges and vertices of Γ splits the sphere into several components, called *cells*. We denote the set of all cells by $\mathcal{C}$. Since Γ is connected, the *boundary* ∂c of a cell c corresponds to a cyclic sequence alternating between edge-segments and elements in $V \cup X$ (i.e., vertices and crossings). If a crossing or a vertex appears multiple times on the boundary of the same cell c, then c is *degenerate*. The *size* of a cell c, denoted by $\|c\|$, is the number of vertex incidences plus the number of edge-segment incidences of c. Note that incidences with crossings are not taken into account. Figure 4 shows all types of cells of size at most 5. Note that, as we forbid two adjacent crossing edges and two edges crossing more than once, all cells of size at most 5 in our drawings are non-degenerate. For $a \in \mathbb{N}$, we denote by $\mathcal{C}_a = \{c \in \mathcal{C} \colon \|c\| = a\}$ the set of all cells of size a.

▶ **Theorem 4** (Density Formula [9])**.** *If Γ is a connected drawing with at least one edge and t is a real number, then*

$$|E| = t(|V| - 2) - \sum_{c \in \mathcal{C}} \left(\frac{t-1}{4} \|c\| - t \right) - |X|.$$

To apply the Density Formula, we count the cells of different sizes. We distinguish several types of cells based on their size and boundary and denote these by small pictograms, such as or . We call a cell *large* if it has size at least 6 and write for this type of cells. By abuse of notation, we denote the number of cells of a certain type by their pictogram.

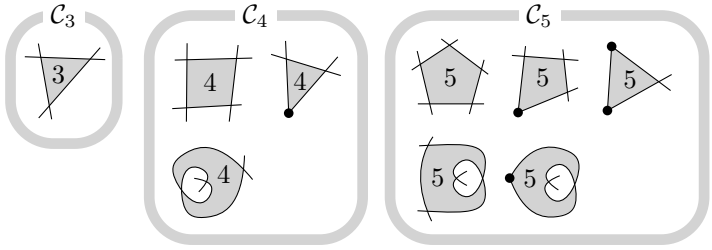


Figure 4 Taken from [9, Figure 2]. All types of cells c of size $\|c\| \leq 5$ in a non-homotopic connected drawing on at least three vertices. The bottom row shows the degenerate cells. Since we consider only simple drawings in this paper, degenerate cells do not appear.

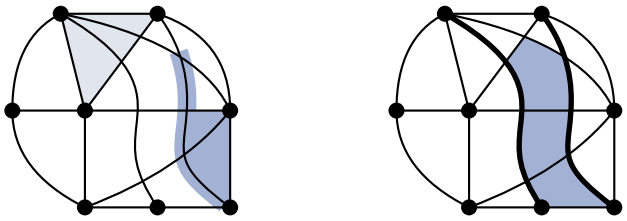


Figure 5 Left: A -configuration (light blue) and a -configuration (dark blue). Right: A -trail (dark blue) and its bounding edges (thick).

Configurations and Trails. Configurations are arrangements of several cells and edge-segments in a drawing Γ of a graph G. Formally, a *configuration* is a connected labeled embedded subgraph H of the planarization Λ of Γ. The label of a vertex $v \in V(H)$ indicates whether v arises from a vertex of G or a crossing. Two configurations are of the same *type* if they are isomorphic as labeled embedded graphs (up to reflection). We also denote the types by small pictograms such as and , see Figure 5 (left) for an example.

A cell c is *interior* to a configuration C if all of its boundary ∂c is part of C. We color interior cells gray in the pictograms. An edge-segment is *interior* to C if both its incident cells are interior to C. For example, a -configuration C has an interior -cell and an interior -cell, whose shared outer edge-segment is the only interior edge-segment of C.

Now, trails are specific configurations. Consider a sequence $(A_1, \ldots, A_\ell)$ of $\ell \geq 2$ cells. The configuration T that is the union of $A_1, \ldots, A_\ell$ is a *trail* if

- neither A_1 nor A_ℓ is a -cell,
- for each $i \in [\ell - 1]$ cells A_i and A_{i+1} share exactly one inner edge-segment (This edge-segment is then interior to T.), and
- if $\ell \geq 3$, then each of $A_2, \ldots, A_{\ell-1}$ is a -cell c whose two interior edge-segments in T are opposite on the boundary ∂c of c.

See Figure 5 (right) for an example. Note that only two of the four inner edge-segments of each -cell of T are interior to T (as no edge self-intersects). Every trail T has two *bounding edges* e_1 and e_2, which are those crossing all interior edge-segments of T. An A-B-trail is a trail whose endpoints are cells of type A and B respectively. We denote by $(A \leftrightarrow B)$ the number of A-B-trails in Γ. Note in particular that $(A \leftrightarrow B) = (B \leftrightarrow A)$.

▶ **Observation 5.** *Every inner edge-segment of a drawing is interior to exactly one trail.*

3-Saturated Drawings. Considering a 3-plane drawing, it is convenient to assume that any two vertices are connected by an uncrossed edge whenever this is obviously possible, i.e., whenever both vertices appear on a common cell. A drawing is *filled* if for every cell c

and every pair of vertices $u \neq v$ on the boundary ∂c of c, there exists an uncrossed edge uv in ∂c. In other words, a drawing Γ is filled if adding any uncrossed edge to Γ results in a lens bounded by two parallel uncrossed edges. For brevity, we define a 3-*saturated* drawing to be a 3-plane, non-homotopic, connected, filled drawing of a graph on at least three vertices. Note that a 3-saturated drawing may contain parallel edges, but also recall that all our drawings are simple.

In every 3-saturated drawing, all cells of size at most five are of one of the following types:  ,  ,  ,  ,  , and  ; cf. Figure 4.

▶ **Lemma 6.** *Every* 3*-plane drawing* Γ *of a connected graph* G *on at least three vertices can be completed to a* 3*-saturated drawing by only adding edges.*

Proof. Note that the drawing Γ is connected as G is connected. We therefore only need to enforce the filled property, while preserving all other properties. Suppose there is a cell c with vertices u and v on its boundary, but no edge uv lies on ∂c. We can insert an edge uv within the cell c, without creating new crossings. The edge might be a parallel edge. Yet, as no edge connecting u and v lies on the boundary ∂c, we did not create an empty lens. Therefore, the drawing is still non-homotopic. Neither does this create self-crossings or crossings between adjacent edges. Inductively, the claim follows. ◀

By Lemma 6, it suffices to consider only 3-saturated drawings in order to obtain upper bounds on the number of crossings and edges in any 3-plane drawing.

3 Bounds on Edge-Density and Crossing-Number of 1-, 2- and 3-Plane Drawings

To obtain our upper bounds we prove a number of (in)equalities, each relating the number of certain cells, configurations, edges and crossings. The Density Formula is one such equality. In total, we obtain a system of linear inequalities where each quantity (such as $|E|$, $(|V|-2)$, $|X|$, $|\mathcal{C}_2|$, $|E_1|$, ,  , etc.) can be considered as a non-negative variable. Setting the "variable" $(|V|-2)$ to 1, we can maximize the value of $|X|$ by solving the obtained linear program (LP). The resulting maximum represents the number of crossings per vertex; more precisely, per $(|V|-2)$.

Suppose we want to prove that the number of crossings in any drawing of a certain type (here we consider 1-, 2-, and 3-plane drawings) on n vertices does not exceed $k(n-2)$ for some fixed k. For this, it suffices to show under the constraints of the LP (where we have normalized $(|V|-2) = (n-2)$ to 1) that the maximum value of the "variable" $|X|$ is at most k. In fact, from what is known as the *dual* of the LP, we obtain a coefficient for each of the LP's inequalities such that multiplying each inequality with the corresponding coefficient and summing the inequalities up yields the desired bound $|X| \leq k(|V|-2)$.

Since we believe that this technique, which may be applicable to other types of drawings, is of independent interest, we first illustrate it on two known examples, namely bounding the crossing number in 1-plane and in 2-plane drawings, and only then turn to 3-plane drawings.

Crossing Number of 1-Plane Drawings. We first illustrate the method on the following known bound for 1-plane drawings.

▶ **Observation 7** ([17, Proposition 4.4])**.** *Every* 1*-plane drawing* Γ *of a graph* G *on* n *vertices with* $n \geq 3$ *contains at most* $n-2$ *crossings.*

In order to prove Observation 7, we first derive two inequalities relating the number of crossings, edges and vertices in 1-plane drawings. Let Γ be a 1-plane drawing on at least three vertices. Removing uncrossed edges and considering the resulting components separately, we may assume that Γ is connected and only contains crossed edges. We may further assume that Γ contains at least three vertices as this holds as soon as we have one crossing. The characterization of cells of small size [9] (cf. Figure 4) implies that Γ contains only cells of size at least 6. Thus, the Density Formula for $t = 3$, (cf. Theorem 4) simplifies to

$$|X| = 3(|V| - 2) - \sum_{c \in \mathcal{C}, \|c\| \geq 6} \left(\frac{1}{2}\|c\| - 3 \right) - |E| \leq 3(|V| - 2) - |E|. \tag{0.A}$$

Moreover, using the fact that every edge in Γ has at most one crossing (in fact, exactly one crossing), a simple double counting argument yields

$$2|X| \leq |E|. \tag{0.B}$$

Observe that (0.A) and (0.B) can be considered as homogeneous linear inequalities in the variables $|X|, |E|, |V| - 2$ and thus any set of values for these variables that satisfy the inequalities can be scaled arbitrarily. Hence, by normalizing $|V| - 2$ to 1, we transform the set of inequalities (left) into an LP with non-negative variables represented by the matrix M_1 (right).

$$\begin{cases} |X| + |E| \leq 3(|V| - 2), & \text{(0.A)} \\ 2|X| - |E| \leq 0, & \text{(0.B)} \end{cases} \quad \longleftrightarrow \quad M_1 = \begin{array}{c} \\ \text{(0.A)} \\ \text{(0.B)} \end{array} \begin{array}{c} \begin{array}{ccc} |X| & |E| & |V|-2 \end{array} \\ \left(\begin{array}{cc|c} 1 & 1 & 3 \\ 2 & -1 & 0 \end{array}\right) \end{array}$$

We want to determine an upper bound on $|X|$ in terms of $|V| - 2$ solely by multiplying the inequalities (0.A) and (0.B) with constant coefficients and summing up. This corresponds to maximizing $|X|$ in the LP. It turns out that in the LP represented by M_1, the maximum value of the "variable" $|X|$ is 1. Computing an optimal solution of the dual LP, we obtain a vector c_1 such that all entries of the vector $s_1 = c_1 \cdot M_1$ are non-negative and the entry corresponding to the "variable" $|X|$ is 1, i.e., the maximum possible value of $|X|$ under the given constraints. Here, we obtain for $c_1 = \left(\frac{1}{3} \quad \frac{1}{3}\right)$ the following.

$$s_1 = c_1 \cdot M_1 = \left(\tfrac{1}{3} \quad \tfrac{1}{3}\right) \begin{array}{c} \begin{array}{ccc} |X| & |E| & |V|-2 \end{array} \\ \left(\begin{array}{cc|c} 1 & 1 & 3 \\ 2 & -1 & 0 \end{array}\right) \end{array} = \begin{array}{c} \begin{array}{ccc} |X| & |E| & |V|-2 \end{array} \\ \left(\begin{array}{cc|c} 1 & 0 & 1 \end{array}\right) \end{array}.$$

The resulting vector $s_1 = \left(\begin{array}{cc|c} 1 & 0 & 1 \end{array}\right)$ can now be reinterpreted as the inequality $|X| \leq |V| - 2$. In fact, the entries of the vector c_1 correspond to the coefficients with which we need to multiply the inequalities (0.A) and (0.B) such that summing up yields $|X| \leq |V| - 2$. This concludes the proof of Observation 7.

Crossing Number of 2-Plane Drawings. In [9], the Density Formula is used to show that every 2-plane drawing of a graph on n vertices contains at most $5n - 10$ edges. The statement follows from a lower bound on the number of crossings (cf. Lemma 9 (1.A)). From the upper bound on the number of edges, the authors of [3] derive an upper bound on the number of crossings in 2-plane drawings.

▶ **Theorem 8** ([3, Theorem 3]). *Every 2-plane drawing Γ of a graph G on n vertices with $n \geq 3$ contains at most $3.\overline{3}(n - 2)$ crossings.*

They prove that at most two thirds of the edges of a filled 2-plane drawing are crossed (cf. Lemma 9 (1.B)). We reformulate the proof based on our method. To this end, we first collect several observations about local interactions of cells, crossings, and edges.

▶ **Lemma 9** ([9, Lemma 8.2, Lemma 3.1], [3, Theorem 3]). *For every connected, non-homotopic 2-plane drawing Γ on $n \geq 3$ vertices, we have*

$$3\,\triangledown + 2\,\square + \pentagon \leq |X|, \tag{1.A}$$

$$|E_\times| \leq 0.\overline{6}|E|, \tag{1.B}$$

$$|E| \leq 5(|V|-2) + 2\,\triangledown + \square + \pentagon - |X|, \tag{1.C}$$

$$2|X| \leq 2|E_\times|. \tag{1.D}$$

Proof. For a proof of (1.A), we refer to [9, Lemma 8.2], a proof of (1.B) is given in [3, Theorem 3]. The constraint (1.C) follows from the Density Formula for $t = 5$ (cf. Theorem 4). To prove (1.D), observe that every edge is crossed at most twice and every crossing is incident to two distinct edges in any 2-plane drawing. A simple double-counting argument now yields (1.D). ◀

Normalizing the variable corresponding to $|V| - 2$ in the inequalities (1.A),(1.B),(1.C) and (1.D) yields an LP with non-negative variables that can be represented by the matrix M_2 with

$$M_2 = \begin{array}{c} \begin{matrix} |X| & |E| & |E_\times| & \triangledown & \square & \pentagon & |V|-2 \end{matrix} \\ \left(\begin{array}{cccccc|c} -1 & 0 & 0 & 3 & 2 & 1 & 0 \\ 0 & -2 & 3 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -2 & -1 & -1 & 5 \\ 2 & 0 & -2 & 0 & 0 & 0 & 0 \end{array}\right) \end{array} \begin{array}{l} \text{(1.A)} \\ \text{(1.B)} \\ \text{(1.C)} \\ \text{(1.D).} \end{array}$$

Under the given constraints, the maximum value of the "variable" $|X|$ is $\frac{10}{3}$. In fact, solving the dual LP, we obtain the vector $c_2 = \left(\frac{2}{3} \quad \frac{1}{3} \quad \frac{2}{3} \quad \frac{1}{2}\right)$ that satisfies both $c_2 \geq 0$ and $c_2 \cdot M_2 = s_2$ for $s_2 = \left(1 \quad 0 \quad 0 \quad \frac{2}{3} \quad \frac{2}{3} \quad 0 \;\middle|\; \frac{10}{3}\right)^{\mathrm{T}}$. The vector s_2 can now be reinterpreted as the inequality

$$|X| + \frac{2}{3}\,\triangledown + \frac{2}{3}\,\square \leq \frac{10}{3}(|V|-2). \tag{1.E}$$

As the quantities $\triangledown$ and $\square$ are non-negative, we obtain $|X| \leq 3.\overline{3}(|V|-2)$, which concludes the proof of Theorem 8. Again, note that summing up (1.A)–(1.D) with the coefficients from c_2 yields precisely (1.E). Interpreting a set of inequalities as an LP enables us to easily compute a best-possible bound and the dual coefficients to witness it.

Crossing Number of 3-Plane Drawings. We derive the inequalities for 3-plane drawings in Section 4. Our LP comprises 21 constraints, which are summarized in Figure 13 in the appendix, and whose validity is proven in Section 4. Summing up all constraints using the coefficients shown in Figure 13, we obtain $|X| \leq 5.5(|V|-2)$.

If we maximize $|E|$ instead of $|X|$ in the LP, we obtain $|E| \leq 5.5(|V|-2)$ from the same constraints (witnessed by different coefficients; also in Figure 13). Hence, by verifying that all 21 constraints hold for every 3-saturated drawing, we obtain our result by Lemma 6.

As in the case of 1-plane drawings, the LP for 3-plane drawings can be modeled as a matrix M where each column corresponds to a non-negative variable v_i and each row to an inequality of the form $\sum_i a_i v_i \leq b(|V|-2)$ with constant, integer factors a_i and b. Solving

the dual LP, we obtain a vector $c_\times$ such that $s_\times = c_\times \cdot M$ is a vector where each entry is non-negative and the entry corresponding to the number of crossings $|X|$ is 1, the one corresponding to $|V| - 2$ is 5.5. This witnesses $|X| \leq 5.5(|V| - 2)$. Similarly, there exists a vector c_E such that $c_E \cdot M$ yields $|E| \leq 5.5(|V| - 2)$. The explicit definition of M, $c_\times$ and c_E is given in Appendix A.

4 Relating Crossing, Edge, Cell, Trail, and Configuration Counts

In this section, we present a number of (in)equalities, each relating the number of certain cells, configurations, edges, or crossings. Our proof relies on the Density Formula for $t = 5$. For this value of t, ◯-cells contribute negatively in the formula. Intuitively, large cells account for many crossings: If many trails end in large cells, we obtain a lower bound on the sum $\sum_{a \geq 6} a|\mathcal{C}_a|$ of sizes of large cells. This yields a lower bound on the sum $\sum_{c \in \mathcal{C}_{\geq 6}} (\|c\| - 5)$ in the Density Formula, where $\mathcal{C}_{\geq 6}$ denotes the set of large cells. If there are few such trails, we obtain configurations that contain many crossed edges.

4.1 Lower Bounds on the Number of Cells and Configurations

▶ **Lemma 10.** *If Γ is a 3-saturated drawing, then*

$$\text{(4)} = (\text{(4)} \leftrightarrow \text{(5)}) + (\text{(4)} \leftrightarrow \text{(5)}) + (\text{(4)} \leftrightarrow \bigcirc) \tag{2.A}$$

$$2\,\text{(5)} = (\text{(4)} \leftrightarrow \text{(5)}) + 2\,(\text{(5)} \leftrightarrow \text{(5)}) + (\text{(5)} \leftrightarrow \text{(3)}) + (\text{(5)} \leftrightarrow \text{(5)}) + (\text{(5)} \leftrightarrow \bigcirc) \tag{2.B}$$

$$3\,\text{(3)} = (\text{(3)} \leftrightarrow \text{(5)}) + (\text{(3)} \leftrightarrow \text{(5)}) + (\text{(3)} \leftrightarrow \bigcirc) \tag{2.C}$$

$$5\,\text{(5)} = (\text{(3)} \leftrightarrow \text{(5)}) + (\text{(5)} \leftrightarrow \text{(4)}) + (\text{(5)} \leftrightarrow \text{(5)}) + 2\,(\text{(5)} \leftrightarrow \text{(5)}) + (\text{(5)} \leftrightarrow \bigcirc) \tag{2.D}$$

Proof. We only prove the first equality. The remaining equations can be shown in a similar way. Consider a (4)-cell c. By Observation 5, the inner edge-segment on the boundary of c is interior to a trail starting in c. Let c' be the cell corresponding to the other end of the trail. As parallel edges do not form an empty lens, c' cannot be a (4)-cell. Further, no two adjacent edges cross. Thus, c' cannot be a (3)-cell. Recall that no end of a trail corresponds to a (4)-cell, yet the cell c' has to be incident to an inner edge-segment. Thus, c' is a (5)-, a (5)- or a ◯-cell (i.e., a cell of size at least 6). As every (4)-cell is only incident to one inner edge-segment, we obtain one trail for each such cell. Double-counting yields (2.A). ◀

▶ **Lemma 11.** *If Γ is a 3-saturated drawing, then*

$$(\text{(3)} \leftrightarrow \text{(5)}) \leq \text{(5)}. \tag{3.A}$$

Proof. Consider a (3)-(5)-trail T, see Figure 6. We use the same notation as in Figure 6. Note that T contains no (4)-cell as each of its bounding edges $u'u$ and $v'v$ is already crossed three times. For the same reason, one of the endpoints of uu' and vv' lies on the boundary of c and c', respectively, i.e., $u \in \partial c$ and $v \in \partial c'$. As the edge e is crossed at least twice, an endpoint w of e is incident to c or c'. We may assume without loss of generality that w lies on $\partial c'$. Observe that w and v are both incident to c'. Since the drawing is 3-saturated, $vw \in E(G)$ and c' is a (6)-cell. Similarly, we see that $uv \in E(G)$ and the cell c'' is a (6)-cell. Thus, the cells c' and c'' form a (5)-configuration C.

We now observe that no other (3)-(5)-trail is adjacent to C. The two outer segments of the (6)-cell in C are part of the two bounding edges of the (3)-(5)-trail T, see Figure 6 (right). As no two (3)-(5)-trails have the same bounding edges, each is adjacent to a different (5)-configuration and the inequality above follows. ◀

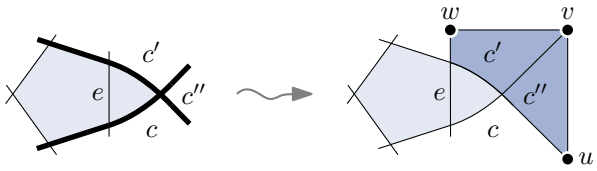


Figure 6 A $\{3\}$-$\{5\}$-trail (light blue) and its boundary (thick) is represented on the left. Such a trail is always adjacent to a $\{5\}$-$\{5\}$ (dark blue). The adjacency is represented on the right.

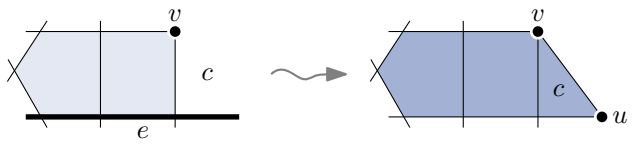


Figure 7 A $\{5\}$-$\{5\}$-trail (light blue). The bounding edge e (thick) is crossed three times. Such a trail forms a $\{5\}$-$\{5\}$ with an adjacent cell. The $\{5\}$-$\{5\}$ (dark blue) is represented on the right.

▶ **Lemma 12.** *If Γ is a 3-saturated drawing, then*

$$(\{5\} \leftrightarrow \{5\}) \leq \{5\}\text{-}\{5\}. \tag{3.B}$$

Proof. Consider a $\{5\}$-$\{5\}$-trail. As the drawing is 3-plane, the trail contains no $\{4\}$-cell, and we are in the situation represented in Figure 7. The edge e (represented by a thick line) is crossed three times. Thus, one of its endpoints lies on the boundary of the cell c. Let u denote this vertex. Note that the vertex v, which is the only vertex on the $\{5\}$-cell, is also incident to c. As the drawing is 3-saturated, uv is an edge and c is a $\{5\}$-cell. The $\{5\}$-$\{5\}$-trail together with c forms a $\{5\}$-$\{5\}$. For every $\{5\}$-$\{5\}$-trail, we obtain a distinct $\{5\}$-$\{5\}$ (which may share a $\{5\}$-cell with another such configuration) and the inequality follows. ◀

▶ **Lemma 13.** *If Γ is a 3-saturated drawing, then*

$$(\{3\} \leftrightarrow \{5\}) \leq \{3\}\text{-}\{5\}\text{-}\{5\}. \tag{3.C}$$

Proof. Consider a $\{3\}$-$\{5\}$-trail. As every edge is crossed at most three times, the trail contains no $\{4\}$-cell and we are in the situation represented in Figure 8. The vertices u and v lie on the boundary of a cell c. As the drawing is 3-saturated, the edge uv is contained in G and the cell c is a $\{5\}$-cell. The trail together with c forms a $\{3\}$-$\{5\}$-$\{5\}$-configuration. As every $\{3\}$-$\{5\}$-trail is only part of one such configuration, the lower bound on the number of $\{3\}$-$\{5\}$-$\{5\}$-configurations follows. ◀

▶ **Lemma 14.** *If Γ is a 3-saturated drawing, then*

$$\{4\} \leq \{5\}\text{-}\{5\} + \{5\}\text{-}\{5\}. \tag{3.D}$$

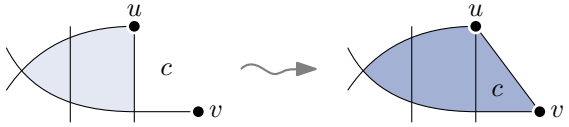


Figure 8 A $\{3\}$-$\{5\}$-trail (light blue). Such a trail forms a $\{3\}$-$\{5\}$-$\{5\}$-configuration (dark blue) with an adjacent cell.

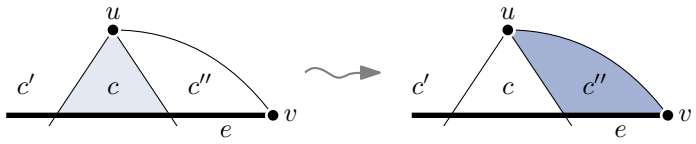


Figure 9 A △-cell c (light blue). At least one of the two cells sharing an outer edge-segment with c is a ▽-cell (dark blue).

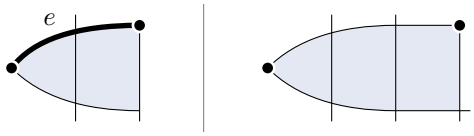


Figure 10 A △-□-trail with no □-cell is depicted on the left, and with one □-cell on the right.

Proof. We show that each △-cell is contained in a +△-configuration or a ▽△-configuration. Consider the inner edge-segment of a △-cell c and let e be the corresponding edge. We denote the two cells sharing an outer edge-segment with c by c' and c''. As the edge e is crossed at most three times, an endpoint v of e lies on the boundary of c' or c''. Without loss of generality, we are therefore in the situation represented in Figure 9. The two vertices u and v lie on the boundary of the cell c''. As Γ is 3-saturated, uv is an edge of G and c'' is a ▽-cell.

If e is crossed three times, the three cells c', c and c'' are in a +△-configuration. Otherwise, e is crossed twice and the same argument as above shows that c' is also a ▽-cell. In particular, the cells c', c and c'' are in a △△-configuration.

As every △-cell is only contained in one such configuration, the upper bound on the number of △-cells follows. ◀

Recall that E_1 denotes the set of edges that have exactly one crossing in the drawing.

▶ **Lemma 15.** *If Γ is a 3-saturated drawing, then*

$$(\triangle \leftrightarrow \square) \leq \frac{1}{2}|E_1| + \square. \tag{3.E}$$

Proof. Note that it suffices to show that every △-□-trail forms a □-configuration or has a bounding edge that is crossed at most once. The claim then follows as every edge is a bounding edge of at most two trails.

Consider a △-□-trail. As every edge is crossed at most three times, the trail contains at most one □-cell.

If the trail contains no such cell, we are in the situation depicted on the left of Figure 10. Note that the bounding edge e of the trail is crossed only once.

If the trail contains a □-cell, we are in the situation represented on the right of Figure 10. The trail forms a □-configuration.

Thus, the number of △-□-trails yields a lower bound on the sum of half the number of edges that are crossed only once, and the number of □-configurations. ◀

▶ **Lemma 16.** *If Γ is a 3-saturated drawing, then*

$$2\,(⬠ \leftrightarrow ⬠) + (\triangle \leftrightarrow ⬠) + (▽ \leftrightarrow ⬠) - 4\,⬠ \leq ⬠. \tag{4.A}$$

Proof. Let T be a trail ending in a ⬠-cell and let e_1 and e_2 be the two edges on its boundary. If the other end of T corresponds to a △-, ⬠- or ▽-cell, the edges e_1 and e_2 have the same number of crossings within the trail. We therefore call these cells *crossing-even* and use the



Figure 11 Left: A full 5-cell with its incident trails is represented. One of the trails is crossed. The corresponding boundary edges are represented with thick lines. Right: A full 5-cell with three consecutive uncrossed trails.

same term for trails where one endpoint corresponds to a 5-cell, the other to a crossing-even cell. A 5-cell where every incident trail ends in a crossing-even cell is called *full*. We denote the number of full 5-cells by ℓ.

An upper bound on the number ℓ of full 5-cells. We first show that every full 5-cell is part of at least one -configuration. As no such configuration contains two 5-cells, we then obtain

$$\ell \leq .\qquad (\star)$$

Consider a full 5-cell c. We say that a trail incident to c is *uncrossed* if its boundary contains no crossing that is not incident to c. Otherwise, we say that it is *crossed*. In fact, all such uncrossed trails end in a 4-cell and contain no 4-cell. Every trail T incident to c contains an inner edge-segment $s(T)$ on the boundary of c in its interior. Two such trails T and T' are called *consecutive* if the edge-segments $s(C)$ and $s(C')$ share a crossing.

We first show that there are at least three consecutive uncrossed trails incident to the full cell c. If none of the incident trails of c is crossed, the claim clearly holds. Otherwise, some trail A is crossed, see Figure 11 (left). Recall that A is crossing-even. Thus, the two edges on its boundary are crossed three times. Therefore, the trails C and D are uncrossed. The same argument shows that only one of the two remaining trails B and E is crossed. We therefore obtain three consecutive uncrossed trails.

As every uncrossed trail ends in a 4-cell and contains no 4-cell, we are in the situation depicted in Figure 11 (right). Note that the vertices u and v lie on the boundary of a cell c'. Thus, $uv \in E(G)$ and c' is a 5-cell as Γ is 3-saturated. Similarly, we see that $vw \in E(G)$ and c'' is a 5-cell. The cells c, c' and c'' form a -configuration. It follows that every full 5-cell is part of a -configuration.

A lower bound on the number ℓ of full 5-cells. We define

$$k := 2\left(\leftrightarrow\right) + \left(\leftrightarrow\right) + \left(\leftrightarrow\right),$$

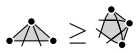
i.e., k corresponds to the number of incidences between crossing-even trails and 5-cells. It remains to show that

$$k - 4 \cdot 5 \leq \ell.\qquad (\star\star)$$

The claim then follows by combining $(\star)$ and $(\star\star)$.

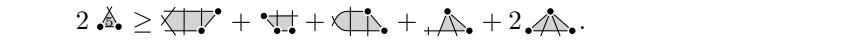
Every  -cell that is not full is incident to at most four crossing-even trails while every full  -cell is incident to five such trails. As there are k incidences between crossing-even trails and  -cells, we obtain $k \leq 4(\text{} - \ell) + 5\ell = 4\text{} + \ell$. Thus, ($\star\star$) holds. ◀

▶ **Lemma 17.** *If* Γ *is a* 3*-saturated drawing, then*

 (4.B)

Proof. Each  -configuration contains a  . As no two  -configurations share a  , the claim follows. ◀

▶ **Lemma 18.** *For every drawing* Γ*, we have*

 (9.C)

Proof. We give a lower bound on the number of -cells by double-counting the number of outer edge-segments incident to a -cell.

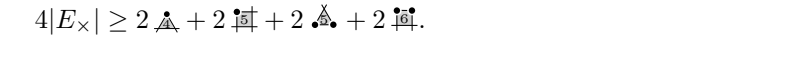
Every -cell is incident to two outer edge-segments.

Recall that an edge-segment is interior to a configuration C if both its incident cells are interior to C. In each of the configurations  ,  ,  ,  and  , exactly one outer edge-segment s_c of each interior -cell c is interior to the configuration. Note in particular that we did not count any such edge-segment twice due to the choice of the configurations we consider. Indeed, any two configurations may share a -cell c, but the corresponding outer edge-segments s_c do not coincide. As each  ,  ,  and  contains one outer edge-segment that is incident to a -cell in its interior, and  contains two such segments, the lower bound on the number of -cells follows. ◀

4.2 Lower Bounds on the Number of Edges

Recall that $E_\times$ denotes the set of edges with at least one crossing.

▶ **Lemma 19.** *For every drawing* Γ*, we have*

 (6)

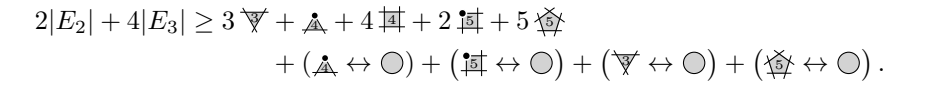
Proof. We double count the number ℓ of incidences between outer edge-segments and cells.

Every crossed edge has exactly two outer edge-segments. Each of them has two incidences to cells. Thus, $4|E_\times| = \ell$.

Every  -cell, every  -cell, every -cell and every  -cell is incident to exactly two outer edge-segments. We therefore have, $\ell \geq 2\text{} + 2\text{} + 2\text{} + 2\text{}$ which yields the claim. ◀

Recall that for an integer $i \geq 0$, we denote by E_i the set of edges with exactly i crossings.

▶ **Lemma 20.** *For every drawing* Γ*, we have*

 (7)

Proof. We double count the number ℓ of incidences between inner edge-segments and cells.

An edge that is crossed k times is split into $k+1$ edge-segments, $k-1$ of which are inner edge-segments. Each inner edge-segment is incident to two cells. We therefore have $2|E_2| + 4|E_3| \geq \ell$.

On the other hand, every  -cell, every -cell, every  -cell, every  -cell and every  -cell is incident to 3, 1, 4, 2 and 5 inner edge-segments respectively. In order to obtain a lower bound for the number of inner edge-segment incidences with large cells, it suffices to count the number of trails ending in large cells. Indeed, each such trail enters a large cell through an inner edge-segment. This yields the right hand side of the inequality above. ◀

▶ **Lemma 21.** *For every drawing* Γ*, we have*

$$|E_2| \geq \frac{1}{2} \cdot \text{  } + \text{  } \tag{8.C}$$

Proof. Every  -configuration contains an edge that is crossed exactly twice. This edge may be contained in at most two such configurations. Similarly, every  -configuration contains an edge that is crossed twice. Note that this edge cannot be contained in any other  - or  -configuration.

Thus, the sum of $\frac{1}{2}$  and  provides a lower bound on the number $|E_2|$ of edges that are crossed exactly twice. ◀

Double-counting the number of incidences of uncrossed edges with cells yields the following.

▶ **Lemma 22.** *For every drawing* Γ*, we have*

$$2|E_0| \geq \text{  } + \text{  }. \tag{9.B}$$

4.3 Relating the Number of Cells, Crossings and Edges

Let us start with a convenient consequence of the Density Formula with parameter $t = 5$. We write $\mathcal{C}$ for the set of all cells. For $a \in \mathbb{N}$, we denote by $\mathcal{C}_a$ the set of all cells of size a.

▶ **Corollary 23.** *If* Γ *is a* 3*-saturated drawing on at least three vertices, then*

$$|X| \leq 5|V| + 2 \cdot \text{} + \text{  } + \text{  } - \frac{1}{6} \sum_{a \geq 6} a|\mathcal{C}_a| - |E|. \tag{5.B}$$

Proof. Recall that every 3-saturated drawing is in particular connected. As the graph contains at least three vertices, every cell has size at least 3. Considering $t = 5$ in the Density Formula (Theorem 4) yields

$$|E| \leq 5|V| - \sum_{c \in \mathcal{C}} (\|c\| - 5) - |X|. \tag{$\star$}$$

Thus, only cells of size at most 4 have a positive contribution to the right side. Recall that there is only one cell-type of size 3, namely  , and only two types of size 4:  and  . Cells of size 5 have no contribution. For $a \geq 6$, we have $a - 5 \geq \frac{1}{6}a$. We therefore obtain

$$\sum_{c \in C, \|c\| \geq 6} (\|c\| - 5) = \sum_{a \geq 6} |\mathcal{C}_a|(a-5) \geq \frac{1}{6} \sum_{a \geq 6} a|\mathcal{C}_a|.$$

Together with ($\star$), we get

$$|E| \leq 5|V| + 2 \cdot \text{} + \text{  } + \text{  } - \frac{1}{6} \sum_{a \geq 6} a|\mathcal{C}_a| - |X|. \quad \blacktriangleleft$$

▶ **Lemma 24.** *If* Γ *is a* 3*-saturated drawing, then*

$$\sum_{a\geq 6} a|\mathcal{C}_a| \geq (\text{△} \leftrightarrow \bigcirc) + (\text{5} \leftrightarrow \bigcirc) + (\text{▽} \leftrightarrow \bigcirc) + (\text{5} \leftrightarrow \bigcirc) + 5\,\text{6}. \tag{5.A}$$

Proof. As we want to obtain a lower bound on the sum $\sum_{a\geq 6} a|\mathcal{C}_a|$, it suffices to count the number of vertex and edge-segment incidences of large cells. Each trail that ends in a large cell enters this cell via an inner edge-segment. As no two trails share such an inner edge-segment, we obtain one edge-segment incidence for each such trail.

A 6-cell is in particular large. As it is incident to only one inner-segment, it is the endpoint of only one trail. We have not counted the remaining three edge-segment incidences and the two vertex incidences when considering trails. Therefore, each 6-cell yields at least five more edge-segment and vertex incidences. ◀

5 Discussion

Optimal k-planar graphs are k-planar graphs with maximum number of edges. For $k = 2, 3$, they turn out to admit k-plane drawings with maximum number of crossings among all k-plane drawings of n-vertex graphs. For $k = 4$, there exists a family of simple 4-plane drawings with $6n - 18$ edges and $7.5n - 33$ crossings (Figure 12).

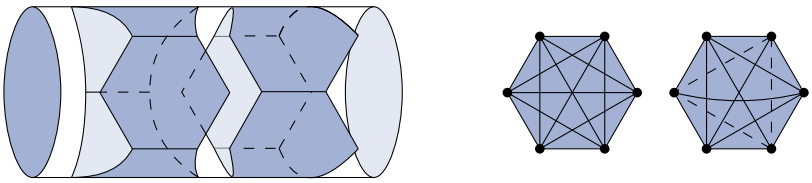


Figure 12 Construction from [1, Figure 35]. Left: A cylinder with two layers, each consisting of three hexagonal faces. Right: To each face of a layer all diagonals are added. To the top and bottom face six diagonals are added. Missing diagonals are represented by dashed lines.

The number of edges $6n - 18$ is optimal for 4-planar graphs up to the additive constant [1, Theorem 4].

▶ **Question 25.** *Is there a constant* c *such that every simple* 4*-plane* n*-vertex drawing contains at most* $7.5n + c$ *crossings?*

The *k-planar crossing number* $\mathrm{cr}_k(G)$ is similar to the crossing number, except that the minimum is taken over all k-plane drawings of G. Clearly, $\mathrm{cr}(G) \leq \mathrm{cr}_k(G)$ for all k and G. But there are k-planar n-vertex graphs G with $\mathrm{cr}(G) \in \mathcal{O}(k)$ and $\mathrm{cr}_k(G) \in \Omega(kn)$ [7, Theorem 2]. By Theorem 1, every 3-plane drawing of an n-vertex graph G has $|X| \leq 5.5(n-2)$ crossings, and hence $\mathrm{cr}(G) \leq \mathrm{cr}_3(G) \leq 5.5(n-2)$. Although Theorem 1 is tight, we could have $\mathrm{cr}(G), \mathrm{cr}_3(G) < 5.5(n-2)$, and a similar question arises for 2-planar graphs. In fact, optimal 3-planar graphs G satisfy $\mathrm{cr}_3(G) = 5.5(n-2)$ [6, Corollary 4] and for optimal 2-planar graphs G we have $\mathrm{cr}_2(G) = 3.\overline{3}(n-2)$ [6, Corollary 2], matching the upper bound in Theorem 8. Yet, $\mathrm{cr}(G)$ might be much smaller.

▶ **Question 26.**

- *Are there* 3*-planar* n*-vertex graphs* G *with* $\mathrm{cr}(G) = 5.5(n-2)$*?*
- *Are there* 2*-planar* n*-vertex graphs* G *with* $\mathrm{cr}(G) = 3.\overline{3}(n-2)$*?*

References

1 Eyal Ackerman. On topological graphs with at most four crossings per edge. *Computational Geometry*, 85:101574, 2019. doi:10.1016/j.comgeo.2019.101574.

2 Martin Aigner and Günter M. Ziegler. *Proofs from THE BOOK*. Springer, Berlin, 6th edition, 2018. doi:10.1007/978-3-662-57265-8_45.

3 Michael A. Bekos, Prosenjit Bose, Aaron Büngener, Vida Dujmović, Michael Hoffmann, Michael Kaufmann, Pat Morin, Saeed Odak, and Alexandra Weinberger. On k-planar graphs without short cycles. In *Proc. 32nd Internat. Sympos. Graph Drawing Network Visualization (GD 2024)*, volume 320 of *LIPIcs*, pages 27:1–27:17, 2024. doi:10.4230/LIPIcs.GD.2024.27.

4 Michael A. Bekos, Michael Kaufmann, and Chrysanthi N. Raftopoulou. On optimal 2-and 3-planar graphs. In *33rd International Symposium on Computational Geometry (SoCG 2017)*, 2017. doi:10.4230/LIPIcs.SoCG.2017.16.

5 Rainer Bodendiek, Heinz Schumacher, and Klaus Wagner. Bemerkungen zu einem Sechsfarbenproblem von G. Ringel. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 53:41–52, 1983. doi:10.1007/BF02941309.

6 Aaron Büngener and Michael Kaufmann. Improving the Crossing Lemma by characterizing dense 2-planar and 3-planar graphs. In *Proc. 32nd Internat. Sympos. Graph Drawing Network Visualization (GD 2024)*, volume 320 of *LIPIcs*, pages 29:1–29:22, 2024. doi:10.4230/LIPICS.GD.2024.29.

7 Markus Chimani, Torben Donzelmann, Nick Kloster, Melissa Koch, Jan-Jakob Völlering, and Mirko H. Wagner. Crossing numbers of beyond planar graphs re-revisited: A framework approach. In *32nd Internat. Sympos. Graph Drawing Network Visualization (GD 2024)*, volume 320 of *LIPIcs*, pages 33:1–33:17, 2024. doi:10.4230/LIPIcs.GD.2024.33.

8 Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani. A survey on graph drawing beyond planarity. *ACM Comput. Surv.*, 52(1):1–37, 2020. doi:10.1145/3301281.

9 Michael Kaufmann, Boris Klemz, Kristin Knorr, Meghana M. Reddy, Felix Schröder, and Torsten Ueckerdt. The density formula: One lemma to bound them all. In *32nd Internat. Sympos. Graph Drawing Network Visualization(GD 2024)*, volume 320 of *LIPIcs*, pages 7:1–7:17, 2024. doi:10.4230/LIPIcs.GD.2024.7.

10 János Pach, Radoš Radoičić, Gábor Tardos, and Géza Tóth. Improving the Crossing Lemma by finding more crossings in sparse graphs. In *Proc. 20th Annu. Sympos. Comput. Geom. (SoCG 2004)*, pages 68–75, 2004. doi:10.1145/997817.997831.

11 János Pach, Radoš Radoičić, Gábor Tardos, and Géza Tóth. Improving the Crossing Lemma by finding more crossings in sparse graphs. *Discrete Comput. Geom.*, 36(4):527–552, 2006. doi:10.1007/s00454-006-1264-9.

12 János Pach, Gábor Tardos, and Géza Tóth. Crossings between non-homotopic edges. In *Proc. 28th Internat. Sympos. Graph Drawing Network Visualization (GD 2020)*, volume 12590 of *LNCS*, pages 359–371, 2020. doi:10.1007/978-3-030-68766-3_28.

13 János Pach, Gábor Tardos, and Géza Tóth. Crossings between non-homotopic edges. *J. Comb. Theory B*, 156:389–404, 2022. doi:10.1016/J.JCTB.2022.05.007.

14 János Pach and Géza Tóth. Graphs drawn with few crossings per edge. In *Proc. 4th Internat. Sympos. Graph Drawing (GD 1996)*, volume 1190 of *LNCS*, pages 345–354, 1996. doi:10.1007/3-540-62495-3_59.

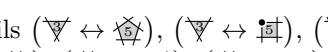
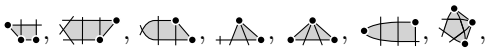
15 János Pach and Géza Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17(3):427–439, 1997. doi:10.1007/BF01215922.

16 Gerhard Ringel. Ein Sechsfarbenproblem auf der Kugel. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 29:107–117, 1965. doi:10.1007/BF02996313.

17 Yusuke Suzuki. 1-planar graphs. In *Beyond Planar Graphs, Communications of NII Shonan Meetings*, pages 47–68. Springer, 2020. doi:10.1007/978-981-15-6533-5_4.

A Certificates

The LP for 3-plane drawings (see Figure 13 for an overview) corresponds to a coefficient matrix M. Each column of M corresponds to a variable v_i and each row to an inequality of the form $\sum_i a_i v_i \leq b(|V|-2)$ with constant, integer coefficients a_i and b. The columns of M correspond (in that order) to

- $|X|$, $|E|$, $|E_0|$, $|E_1|$, $|E_2|$, $|E_3|$, $|E_\times|$,
- the cells  ,
- the trails ,
- the configurations  ,
- and the terms $\sum_{a\geq 6} a|\mathcal{C}_a|$, $|V|-2$.

The rows of M correspond (in that order) to the inequalities (2.A), (2.B), (2.C), (2.D), (3.A), (3.B), (3.C), (3.D), (3.E), (4.A), (4.B), (5.A), (5.B), (6), (7), (8.A), (8.B), (8.C), (9.A), (9.B), (9.C). There is a vector $c_\times$ such that $s_\times = c_\times \cdot M$ is a vector where each entry is non-negative, the entry corresponding to the number of crossings $|X|$ is 1, and the entry corresponding to the number of vertices $|V|-2$ is 5.5. That is, $|X| \leq 5.5(|V|-2)$. Similarly, there exists a vector c_E such that $c_E \cdot M$ yields $|E| \leq 5.5(|V|-2)$. The computation of $c_\times \cdot M$ and $c_E \cdot M$ can be verified with the code given in Listing 1.

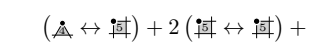
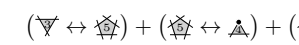
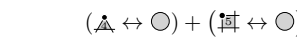
	Inequality	$\lvert E\rvert$	$\lvert X\rvert$
(2.A)		$\frac{-5}{16}$	$\frac{-7}{16}$
(2.B)		$\frac{5}{16}$	$\frac{5}{16}$
(2.C)		$\frac{-11}{24}$	$\frac{-11}{24}$
(2.D)		$\frac{1}{8}$	$\frac{-3}{8}$
(3.A)		$\frac{7}{48}$	$\frac{1}{48}$
(3.B)		0	$\frac{1}{16}$
(3.C)		$\frac{3}{16}$	$\frac{7}{48}$
(3.D)		$\frac{3}{16}$	$\frac{5}{16}$
(3.E)		0	$\frac{1}{16}$
(4.A)		$\frac{3}{16}$	$\frac{13}{16}$
(4.B)		$\frac{3}{16}$	$\frac{5}{16}$
(5.A)		$\frac{11}{60}$	$\frac{11}{60}$
(5.B)		$\frac{11}{60}$	$\frac{11}{60}$
(6)		$\frac{13}{80}$	$\frac{3}{80}$
(7)		$\frac{11}{40}$	$\frac{11}{40}$
(8.A)		$\frac{-11}{20}$	$\frac{19}{20}$
(8.B)		$\frac{11}{20}$	$\frac{1}{20}$
(8.C)		0	$\frac{1}{4}$
(9.A)		$\frac{1}{10}$	$\frac{11}{10}$
(9.B)		$\frac{1}{20}$	$\frac{11}{20}$
(9.C)		$\frac{3}{16}$	$\frac{5}{16}$

Figure 13 Certificates for the upper bound on the number of edges and crossings in 3-saturated drawings in terms of the number of vertices. Each row corresponds to one inequality. In order to obtain the upper bound on the number of edges, we multiply each inequality with the third entry in the corresponding row and sum up all the inequalities. To obtain the upper bound on the number of crossings we proceed likewise using the fourth entry of each row as a coefficient.

Listing 1 Python code for verifying the matrix multiplication of $c_\times \cdot M$ and $c_e \cdot M$.

```
import numpy as np
from fractions import Fraction as fr

ce = [fr(-5,16),fr(5,16),fr(-11,24),fr(1,8),fr(7,48),0,fr(3,16),
fr(3,16),0,fr(3,16),fr(3,16),fr(11,60),fr(11,60),fr(13,80),
fr(11,40),fr(-11,20),fr(11,20),0,fr(1,10),fr(1,20),fr(3,16)]

cx = [fr(-7,16),fr(5,16),fr(-11,24),fr(-3,8),fr(1,48),fr(1,16),fr(7,48),
fr(5,16),fr(1,16),fr(13,16),fr(5,16),fr(11,60),fr(11,60),fr(3,80),
fr(11,40),fr(19,20),fr(1,20),fr(1,4),fr(11,10),fr(11,20),fr(5,16)]

M =
   [[0,0,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,-2,0,0,0,1,0,0,1,0,0,1,0,2,1,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,-3,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,-5,0,0,0,1,0,0,1,0,0,2,1,1,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,-1,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,-1,0,0,0,0],
[0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,-2,0,0,0],
[0,0,0,0,0,0,0,0,0,0,-4,0,0,0,1,0,0,1,0,0,2,0,0,0,0,0,0,0,0,0,0,-1,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,0,1,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,5,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,-1,0],
[6,6,0,0,0,0,0,-12,-6,-6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,30],
[0,0,0,0,0,0,-4,0,0,2,0,2,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,-2,-4,0,3,4,1,5,2,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0],
[0,0,0,1,1,1,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[-2,0,0,1,2,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,-2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,0,0],
[0,-1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,-2,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,-2,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,2,0,0,0,0]]

se = np.matmul(ce,M)
sx = np.matmul(cx,M)

print()
print(f'cx = [',' '.join([format(item,'') for item in cx]),']')
print(f'ce = [',' '.join([format(item,'') for item in ce]),']')
print(f'M = ')
print('\n'.join(['' .join(['{:4}'.format(item) for item in row])
      for row in M]))
print(f'cx * M = [',' '.join([format(item,'') for item in sx]),']')
print(f'ce * M = [',' '.join([format(item,'') for item in se]),']')
```

Structural Parameterizations of k-Planarity

Tatsuya Gima
Hokkaido University, Sapporo, Japan

Yasuaki Kobayashi
Hokkaido University, Sapporo, Japan

Yuto Okada
Nagoya University, Japan

Abstract

The concept of k-planarity is extensively studied in the context of Beyond Planarity. A graph is k-planar if it admits a drawing in the plane in which each edge is crossed at most k times. The local crossing number of a graph is the minimum integer k such that it is k-planar. The problem of determining whether an input graph is 1-planar is known to be NP-complete even for near-planar graphs [Cabello and Mohar, SIAM J. Comput. 2013], that is, the graphs obtained from planar graphs by adding a single edge. Moreover, the local crossing number is hard to approximate within a factor $2-\varepsilon$ for any $\varepsilon > 0$ [Urschel and Wellens, IPL 2021]. To address this computational intractability, Bannister, Cabello, and Eppstein [JGAA 2018] investigated the parameterized complexity of the case of $k = 1$, particularly focusing on structural parameterizations on input graphs, such as treedepth, vertex cover number, and feedback edge number. In this paper, we extend their approach by considering the general case $k \geq 1$ and give (tight) parameterized upper and lower bound results. In particular, we strengthen the aforementioned lower bound results to subclasses of constant-treewidth graphs: we show that testing 1-planarity is NP-complete even for near-planar graphs with *feedback vertex set number at most 3* and *pathwidth at most 4*, and the local crossing number is hard to approximate within *any constant factor* for graphs with *feedback vertex set number at most 2*.

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

Keywords and phrases 1-planar graphs, local crossing number, beyond planarity, parameterized complexity, kernelization

Digital Object Identifier 10.4230/LIPIcs.GD.2025.16

Related Version *Full Version*: `https://arxiv.org/abs/2506.10717`

Funding *Tatsuya Gima*: JSPS KAKENHI Grant Number JP24K23847 and JP25K03077.
Yasuaki Kobayashi: JSPS KAKENHI Grant Numbers JP23K28034, JP24H00686, and JP24H00697.
Yuto Okada: Supported by JST SPRING, Grant Number JPMJSP2125.

Acknowledgements We would like to thank Alexander Wolff for valuable discussions in Würzburg and anonymous reviewers for insightful comments.

1 Introduction

Graph drawing is recognized as an important area in the research of graph theory and algorithms due to various real-world applications. In particular, efficient algorithms for drawing graphs on the plane without edge crossings have received considerable attention from various perspectives. However, to visualize real-world networks, it is necessary to consider drawing *non-planar* graphs in many cases. This research direction is positioned as Beyond Planarity [8, 11, 21], and a significant amount of effort has been dedicated to analyzing their graph-theoretic properties and designing algorithms for drawing these graphs. In this context, the problem of drawing an input graph on the plane with the minimum number of crossings is one of the most well-studied problems. In other words, the problem asks for


© Tatsuya Gima, Yasuaki Kobayashi, and Yuto Okada;
licensed under Creative Commons License CC-BY 4.0

Table 1 The table summarizes our and known results. The columns "unbounded", "parameter", and "$k = 1$" indicate the results when k is taken as input, as parameter, and $k = 1$, respectively.

<table>
<tr><th></th><th>k: unbounded</th><th>k: parameter</th><th>$k = 1$</th></tr>
<tr><td>feedback edge set number</td><td colspan="2">FPT (Thm. 14)</td><td>FPT [2]</td></tr>
<tr><td>feedback vertex set number</td><td colspan="3">NP-complete when fvs $= 2$ (Thm. 5)</td></tr>
<tr><td>treedepth</td><td>W[1]-hard (Thm. 6)</td><td>FPT (Thm. 16) (non-uniform FPT [34])</td><td>FPT [2]</td></tr>
<tr><td>longest induced path</td><td rowspan="2">W[1]-hard (Thm. 13)</td><td colspan="2" rowspan="2">FPT (Cor. 18)</td></tr>
<tr><td>twin cover number</td></tr>
<tr><td>distance to path forest</td><td colspan="3">W[1]-hard (Thm. 9)</td></tr>
<tr><td>domination number</td><td colspan="3">NP-complete when dn $= 2$ (Cor. 10)</td></tr>
</table>

the *crossing number* of an input graph, which is known to be NP-hard [15]. Among various studies on this problem, the (recent) progress on the parameterized complexity of Crossing Number, where the goal is to determine whether an input graph G can be drawn in the plane with at most k crossings, is remarkable [7, 18, 23, 27]. In particular, this problem is *fixed-parameter tractable* (FPT) when parameterized by k, that is, it admits an algorithm with running time $f(k)n^{O(1)}$, where n is the number of vertices in the input graph and f is a computable function.

The *local crossing number* is one of the well-studied variations of the standard crossing number [1, 2, 4, 17, 20, 26, 31, 33], as witnessed by the fact that it is selected as the topic for the live challenge[1] held in conjunction with GD 2025. Let k be an integer. We say that a graph G is *k-planar* if it can be drawn in the plane so that each edge involves at most k crossings. The minimum integer k such that G admits a k-planar drawing is called the *local crossing number* of G, denoted by $\mathrm{lcr}(G)$. The problem of deciding if $\mathrm{lcr}(G) \leq k$ for a given graph G is called k-Planarity Testing. Unlike Crossing Number, it is already NP-complete to decide whether $\mathrm{lcr}(G) \leq 1$ [2, 4, 17, 26]. More strongly, this problem is NP-complete even when the input is restricted to planar graphs with an extra edge [4] or graphs with constant bandwidth [2]. Urschel and Wellens [33] later extended the NP-hardness by showing that k-Planarity Testing is NP-complete for any fixed $k \geq 1$, even if the input is restricted to graphs with local crossing number at most k or at least $2k$. This implies that, unless P = NP, there is no polynomial-time $(2 - \varepsilon)$-approximation for local crossing number for any $\varepsilon > 0$.

To overcome such an intractability, Bannister, Cabello, and Eppstein [2] pursued the parameterized complexity of the case of $k = 1$, namely 1-Planarity Testing, by focusing on *structural parameterizations*. They showed that 1-Planarity Testing is FPT parameterized by treedepth and by feedback edge number (see Section 2 for definitions). As mentioned above, 1-Planarity Testing is NP-complete even on the classes of graphs with bounded bandwidth. This intractability is indeed inherited by wider classes of graphs, such as those with bounded cliquewidth, treewidth, and pathwidth. In this direction, Münch, Pfister, and Rutter [28] recently considered k-Planarity Testing on special cases of pathwidth-bounded graphs and gave exact and approximation algorithms.

[1] https://mozart.diei.unipg.it/gdcontest/2025/live/

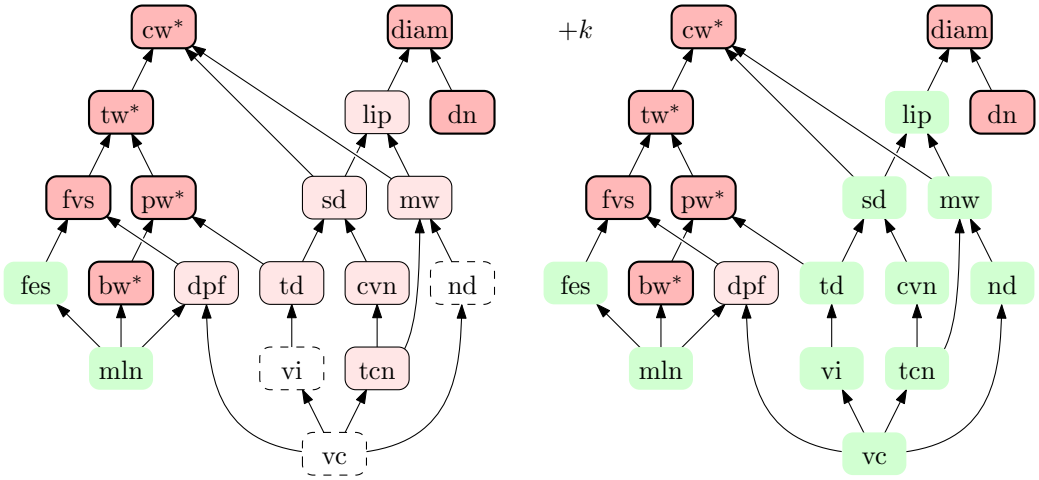


Figure 1 A visualization of Table 1 of the cases where k is (a) unbounded and (b) parameter. No borderline, dotted borderline, normal borderline, bold borderline mean that the complexity parameterized by the parameter is FPT, unknown, W[1]-hard, paraNP-complete, respectively. The complexities for parameters with $(*)$ are already known results and the others are our results. An arrow indicates that the upper parameter is bounded from above by a function of the lower parameter. See the footnote[2] for the abbreviations of parameter names.

In this paper, we extend results of [2] by considering the general case $k \geq 1$ and show a fine parameterized complexity landscape with respect to graph width parameters (see Table 1 and Figure 1 for a summary of our results[2]). We extend algorithms of [2] by showing that k-Planarity Testing is FPT parameterized by feedback edge set number (Theorem 14) and by treedepth plus k (Theorem 16). More generally, we show that k-Planarity Testing is FPT on P_t-free graphs when parameterized by $t+k$ (Corollary 18). We also give polynomial kernelizations for k-Planarity Testing with respect to vertex cover number (Theorem 23) and neighborhood diversity (Corollary 25). On the negative side, we show that k-Planarity Testing is W[1]-hard when parameterized solely by treedepth (Theorem 6), which indicates that Theorem 16 is tight in a certain sense, and by twin cover number (Theorem 13). We also show that 1-Planarity Testing is W[1]-hard parameterized by distance to path forest (Theorem 9). Similar to [4], the last result also proves that 1-Planarity Testing remains NP-complete even on planar graphs with a single additional edge. Our reduction shows that 1-Planarity Testing is NP-complete even when further adding restrictions that the input graph has feedback vertex set number at most 3. Using a completely different reduction, we show that 1-Planarity Testing is NP-complete even if the input is restricted to graphs with feedback vertex set number 2 (Theorem 5). This reduction also shows that there is no constant factor approximation for k-Planarity Testing unless P = NP, which significantly strengthens the $(2-\varepsilon)$-inapproximability result of Urschel and Wellens [33].

Several proofs (marked with $\star$) are omitted and available in the full version.

[2] In Figure 1, the names are abbreviated as follows: cw is clique width, diam is diameter, tw is treewidth, lip is longest induced path, dn is domination number, fvs is feedback vertex set, pw is pathwidth, sd is shrub-depth, mw is modular-width, fes is feedback edge set, bw is bandwidth, dpf is distance to path forest, td is treedepth, cvn is cluster vertex deletion number, nd is neighborhood diversity, mln is max leaf number, vi is vertex integrity, tcn is twin cover number, and vc is vertex cover number.

2 Preliminaries

Let G be a graph. The vertex set and edge set of G are denoted by $V(G)$ and $E(G)$, respectively. For a vertex set $X \subseteq V(G)$ (resp. edge set $X \subseteq E(G)$), $G-X$ denotes the graph obtained from G by deleting all elements in X. The subgraph of G induced by $X \subseteq V(G)$ is denoted by $G[X]$. For $v \in V(G)$, $N_G(v)$ denotes the set of neighbors of v.

A (*topological*) *drawing* Γ of G is a representation in the plane that maps the vertices of G to distinct points in the plane and each edge of G to a non-self-intersecting (Jordan) curve connecting the points corresponding to the endpoints. In the rest of this paper, we may simply refer to these points as vertices and these curves as edges when no confusion is possible. A *crossing* in Γ is an intersection of distinct edges that is not a common endpoint. We assume that, in any drawing, an edge does not pass through any vertex other than its endpoints, and no three (or more) edges cross at a common point. The *crossing number* of G, denoted by $\mathrm{cr}(G)$, is the minimum number of crossings over all possible drawings of G. For an integer k, a drawing Γ is said to be *k-planar* if each edge is crossed at most k times. A graph is said to be k-planar if it admits a k-planar drawing. The *local crossing number* of G, denoted by $\mathrm{lcr}(G)$, is the minimum integer k such that G is k-planar. Throughout this paper, we use the following property of local crossing number.

▶ **Lemma 1 ($\star$).** *Let G be a graph and k be a positive integer. Let G_k be the graph obtained from G by subdividing each edge $k-1$ times. Then, $\mathrm{lcr}(G) \leq k$ if and only if $\mathrm{lcr}(G_k) \leq 1$.*

In this paper, we present parameterized algorithms and hardness results with respect to several graph width parameters. However, some of them may not need to be defined precisely. Hence, we only provide the definitions that are directly needed to present our results. For basic concepts in parameterized complexity, we refer the reader to [5].

Let G be a graph. A *vertex cover* of G is a set of vertices S such that $G-S$ is edge-less, and the *vertex cover number* of G is the minimum size of a vertex cover of G, which is denoted by $\mathrm{vc}(G)$. The *feedback vertex set number* (resp. *feedback edge set number*) of G, denoted by $\mathrm{fvs}(G)$ (resp. $\mathrm{fes}(G)$), is the minimum cardinality of a vertex set (resp. edge set) X such that $G-X$ is acyclic. An *elimination forest* of G is a rooted forest F (i.e., a set of rooted trees) such that $V(F)=V(G)$ and for every edge in G, one of the endpoints is an ancestor of the other in F. The *treedepth* of G, denoted by $\mathrm{td}(G)$, is the minimum integer k such that G has an elimination forest of height k. Two vertices u and v are called *true twins* if they are adjacent and $N_G(u)=N_G(v)$; they are called *false twins* if they are non-adjacent and $N_G(u)=N_G(v)$. The *neighborhood diversity* of G is at most k if the vertex set of G can be partitioned into k sets $V_1,\ldots,V_k$ such that either all the pairs in V_i are true twins or they are false twins. Each set V_i is called a *twin class*. Note that each true twin class induces a clique and each false twin class induces an independent set in G. The neighborhood diversity of G is denoted by $\mathrm{nd}(G)$. A vertex set $S \subseteq V(G)$ is called a *twin cover* of G if each connected component of $G-S$ consists of true twins in G. The *twin cover number* of G is the minimum size of a twin cover of G. The *max leaf number* of G is the maximum number of leaves of a spanning tree of G.

3 Hardness

In this section, we give several intractability results for k-Planarity Testing and 1-Planarity Testing. In particular, we show that k-Planarity Testing is hard to approximate within any constant factor in polynomial time, even on graphs with feedback vertex set number at most 2, and 1-Planarity Testing is NP-complete even on near-planar

graphs with feedback vertex set number 3, where a graph is *near-planar* if it can be obtained from a planar graph by adding a single edge. These two results improve the previous results of [33] and [4]. To this end, we design two completely different reductions, one is given from Two-Sided k-Planarity and the other is given from Unary Bin Packing, which also yield several consequences other than these two results.

3.1 Inapproximability for Graphs with Feedback Vertex Set Number 2

In this subsection, we show that it is NP-hard to approximate the local crossing number of a given graph within any constant factor, even if the graph has feedback vertex set number 2.

Before describing our reductions, we start with a technical lemma, showing that by adding some vertices and edges to a graph, one can impose a certain form on its k-planar drawings.

Let G be a graph with m edges and let k be a positive integer. Let $X \subseteq V(G)$ be a nonempty vertex subset. We define a new graph G' by adding a vertex r and $km+1$ paths of length 2 between r and v for each $v \in X$ to G.[3] We refer to these paths of length 2 as *spokes*.

▶ **Lemma 2** ($\star$)**.** *Suppose that G' has a k-planar drawing Γ. Then, there is a k-planar drawing Γ' of G' such that each spoke has no crossings and the subdrawings of Γ and Γ' induced by G are identical.*

Proof (sketch). Since the edges in G can cross other edges km times in total, for each $u \in X$, there is a spoke between r and u that does not cross any edges of G. Let P_u be such a spoke for each $u \in X$ and $\mathcal{P} \coloneqq \{P_u \mid u \in X\}$. Although spokes in $\mathcal{P}$ may cross, we can untangle them, preserving the subdrawing for G. Then we redraw the other spokes along $\mathcal{P}$. ◀

We will use this lemma in the following form.

▶ **Corollary 3.** *Let $S \subseteq V(G)$. Let G' be a graph obtained from G by adding a vertex s and $k|E(G)|+1$ spokes between s and each $u \in S$. Moreover, let $S' \subseteq V(G')$ and let G'' be a graph obtained from G' by adding a vertex s' and $k|E(G')|+1$ spokes between s' and each $u \in S'$. Suppose that G'' has a k-planar drawing. Then, there is a k-planar drawing of G'' such that all spokes have no crossings.*

Proof. By Lemma 2, G'' has a k-planar drawing Γ' in which all the spokes incident to s' have no crossings. Let Γ be the subdrawing of Γ' induced by G'. As shown in the proof of Lemma 2, for each $u \in S$, there is a spoke P_u between s and u that has no crossing with any edge of G in Γ. Since this spoke P_u does not cross any spoke incident to s', the uncrossing operation and redrawing spokes between s and u used in Lemma 2 never create a new crossing with spokes incident to s'. Thus, by applying Lemma 2 to G' and Γ, we have a k-planar drawing of G'' being claimed. ◀

We now turn to our reduction from Two-Sided k-Planarity, in which we are given a bipartite graph $G = (X \cup Y, E)$ with two independent sets X and Y, and an integer k. The goal is to determine whether G has a *2-layer k-planar drawing*, which is a special case of a k-planar drawing in which the vertices in X are drawn on a line, the vertices in Y are drawn on another line parallel to the first line, and each edge is drawn as a straight line.

[3] The length of a path is defined as the number of edges in it.

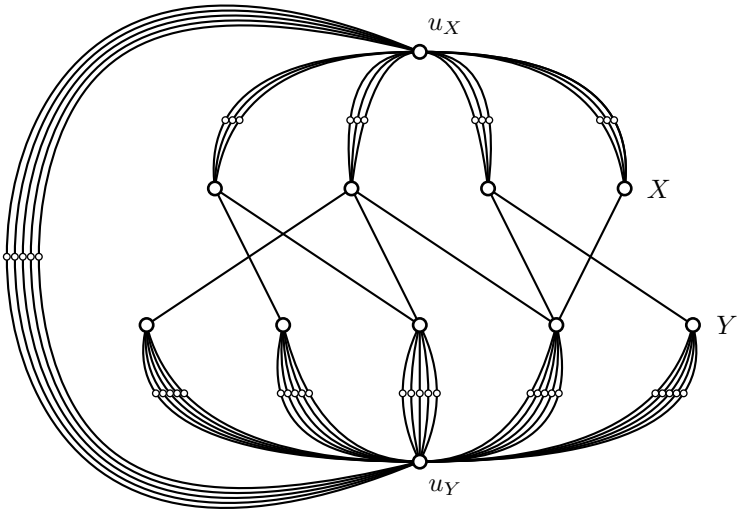


Figure 2 An illustration of the graph G' constructed from an instance $\langle G, k\rangle$ of Two-Sided k-Planarity. Note that the numbers of spokes are different from that of the actual construction.

The idea of our reduction is the same as that of Garey and Johnson [15] from Bipartite Crossing Number to Crossing Number. We remark that, however, the proof is more involved due to the difficulty of k-planarity.

Let $\langle G = (X, Y, E), k\rangle$ be an instance of Two-Sided k-Planarity, where $n_X = |X|$, $n_Y = |Y|$, and $m = |E|$. We construct a graph G' from G as follows (see Figure 2):

1. we introduce two vertices u_X, u_Y;
2. for each $x \in X$, add $\ell_1 \coloneqq km + 1$ spokes between u_X and x;
3. for each $y \in Y \cup \{u_X\}$, add $\ell_2 \coloneqq k(\ell_1 n_X + m) + 1$ spokes between u_Y and y.

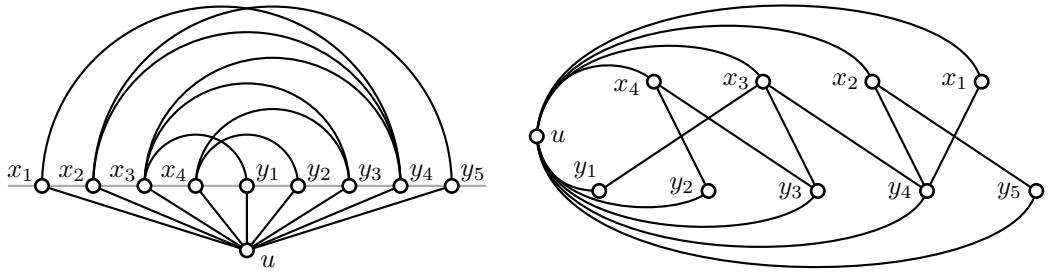
Then, $\langle G', k\rangle$ is the instance we construct for k-Planarity Testing. It is easy to verify that the above construction can be done in polynomial time.

▶ **Lemma 4.** *The graph G admits a 2-layer k-planar drawing if and only if the graph G' admits a k-planar drawing.*

Proof. It is clear that we can obtain a k-planar drawing of G' from a 2-layer k-planar drawing of G by drawing u_X, u_Y, and spokes as Figure 2. Hence, we only show the other direction.

Suppose that the graph G' admits a k-planar drawing Γ'. By Corollary 3, we can assume that all the spokes in G' have no crossings in Γ'. We then replace the set of spokes connecting two vertices with a single edge between them and contract the edge between u_X and u_Y by identifying them into a new vertex u in Γ'. We let G^* denote the graph obtained in this way. As all spokes have no crossings, the obtained drawing of G^*, denoted by Γ^*, is still k-planar. Let us note that G^* is isomorphic to the graph obtained from G by adding a universal vertex u (which is a vertex adjacent to all the vertices in G). We claim that we can draw a closed curve C passing through all the vertices in $V(G^*) \setminus \{u\}$ such that

1. no edge of G^* intersects C (i.e., C is a noose in Γ^*);
2. all the edges incident to u are contained in one of the two regions R with boundary C;
3. all the edges in $E(G)$ are contained in the other region with boundary C;
4. all the vertices in X appear consecutively on C;
5. all the vertices in Y appear consecutively on C.



(a) **(b)**

Figure 3 Two topologically equivalent drawings of G^*, where the subdrawing induced by G is (a) an arc diagram, and (b) a 2-layer drawing.

To this end, we draw a curve C while keeping track of the cyclic ordering of the edges incident to u (and hence the neighbors of u) in Γ^*. As these incident edges have no crossings, we can draw such a curve satisfying (1), (2), and (3). To see the properties (4) and (5), suppose for contradiction that there are $x, x' \in X$ and $y, y' \in Y$ such that x, y, x', y' appear in this order on C. Due to the property (2), C can be seen as a closed curve on Γ' such that the region corresponding to R contains all spokes of G'. Let P_x be a path in G' between x and x' via u_X consisting only of spokes. We define P_y similarly. In Γ' those paths are contained in R and C. However, since the vertices x, x', y, y' are all placed on C, such a drawing must yield a crossing, which contradicts the fact that spokes in G' have no crossings in Γ'.

Let $X = \{x_1, \ldots, x_{n_X}\}$ and let $Y = \{y_1, \ldots, y_{n_Y}\}$ such that $x_1, \ldots, x_{n_X}, y_1, \ldots, y_{n_Y}$ appear on C in this order. We now convert the drawing Γ^* into a drawing $\hat{\Gamma}$ such that

- all the vertices in $V(G^*) \setminus \{u\}$ are placed on a line ℓ in the order $x_1, \ldots, x_{n_X}, y_1, \ldots, y_{n_Y}$;
- all the edges incident to u are drawn as straight segments in a half plane separated by ℓ;
- all the other edges are drawn in the other half plane in such a way that the curve corresponding to an edge forms a semicircular arc between the endpoints.

See Figure 3a for an illustration. Observe that this drawing $\hat{\Gamma}$ is also k-planar as two edges of G^* cross in $\hat{\Gamma}$ only if they cross in Γ^* as well. Moreover, $\hat{\Gamma}$ is *simple*, meaning that no pair of edges crosses more than once and no two edges incident to a common vertex cross. This in turn can be converted into a k-planar drawing Γ as in Figure 3b. The subdrawing of Γ induced by $G^* - \{u\}$ is a 2-layer k-planar drawing such that two edges in $G^* - \{u\}$ cross if and only if they cross in $\hat{\Gamma}$. Hence, G $(= G^* - \{u\})$ admits a 2-layer k-planar drawing. ◀

This reduction immediately gives us the following consequence. Note that Theorem 5 is optimal in the sense that every graph with feedback vertex set number 1 is planar.

▶ **Theorem 5.** 1-Planarity Testing *is NP-complete even if the given graph has feedback vertex set number* 2.

Proof. The NP-membership of 1-Planarity Testing is shown in [26]. It is known that Two-Sided k-Planarity is NP-complete even on trees [25, Theorem 11]. Thus, the graph G' constructed in the proof of Lemma 4 has feedback vertex set number at most 2. This in turn implies the claim by Lemma 1. ◀

The above reduction leads to further consequences. To see this, we first sketch the reduction used in [25, Theorem 11]. They performed a polynomial-time reduction from Bandwidth to Two-Sided k-Planarity. Let T be a tree, b be a positive integer, and ℓ be an arbitrary even number with $\ell \geq 2b^2$. For each edge in T, each edge $e \in E(T)$ is subdivided

once by introducing a new vertex w_e, and we add ℓ leaves adjacent to each original vertex in T. They showed that there is an ordering $v_1, \ldots, v_{|V(T)|}$ of $V(T)$ such that $|i-j| \leq b$ for any edge $\{v_i, v_j\} \in E(T)$ if and only if the obtained graph T' has a 2-layer k-planar drawing with $k = (\ell+4)(b-1)/2$.

▶ **Theorem 6.** *k-Planarity Testing is W[1]-hard with respect to treedepth, even on graphs with feedback vertex set number 2.*

Proof. It is known that the problem Bandwidth is W[1]-hard with respect to treedepth, even on trees [16]. The above reduction we sketched [25, Theorem 11] also preserves the boundedness of treedepth. To see this, we convert an elimination forest F^* of T to that of T' as follows. Starting from F^*, we add the ℓ leaves attached to each vertex as leaf children of it. For each edge $e \in E(T)$, one of the end vertices, say v, is a descendant of the other in F^*. We then add w_e as a leaf child of v in F^*. It is easy to verify that the rooted forest obtained in this way is an elimination forest of T' and its height increases by at most 1 compared to the original elimination forest of T, yielding that $\mathrm{td}(T') \leq \mathrm{td}(T)+1$. Hence, Two-Sided k-Planarity is also W[1]-hard with respect to treedepth, even on trees. Moreover, the reduction used in Lemma 4 increases the treedepth by at most 3, which implies the claim. ◀

▶ **Lemma 7.** *Unless $P = NP$, for any constant $c \geq 1$, there is no polynomial-time c-approximation algorithm for Two-Sided k-Planarity on trees.*

Proof. For any constant $c \geq 1$, it is NP-hard to distinguish the following cases: the bandwidth of T is at most b or more than cb for a given tree T and an integer b [10]. We set $\ell = 2t-4$ for sufficiently large t and construct a tree T' according to the above reduction from T. If the bandwidth of T is at most b, then we have $\mathrm{lcr}(T') \leq t(b-1)$. Otherwise, we have $\mathrm{lcr}(T') \geq tcb$. This implies that a polynomial-time c-approximation algorithm for Two-Sided k-Planarity on trees distinguishes the above two cases. ◀

Combining Lemma 4 and Lemma 7, we obtain the following.

▶ **Theorem 8.** *Unless $P = NP$, for any constant $c \geq 1$, there is no polynomial-time c-approximation algorithm for k-Planarity Testing, even if the given graph has feedback vertex set number 2.*

3.2 Distance to Path Forest

In this subsection, we show that 1-Planarity Testing is W[1]-hard parameterized by distance to path forest, i.e., the minimum number of vertices required to make the graph into a disjoint union of paths. The underlying idea of the proof is similar to the one in [17]. We perform a reduction from Unary Bin Packing. An instance I of Unary Bin Packing consists of a (multi)set of positive integers S and positive integers B and b such that $\sum_{x \in S} x = bB$ and $B = O(\mathrm{poly}(|S|))$. Then the goal is to determine if it is possible to partition S into b sets so that the sum of each set is exactly B. Unary Bin Packing is NP-complete when b is given as input [15] and W[1]-hard when parameterized by b [22].

▶ **Theorem 9.** *1-Planarity Testing is W[1]-hard with respect to distance to path forest.*

Proof. Let $\langle S = \{x_1, \ldots, x_{|S|}\}, B, b\rangle$ be an instance of Unary Bin Packing. In the following, we assume that $b \geq 3$. We construct an instance $\langle G = (V, E)\rangle$ of 1-Planarity Testing as follows. Let G be a graph consisting of vertices $u_1, u_2, v_1, \ldots, v_b$, and paths $P_1, \ldots, P_{|S|}$ with $|V(P_i)| = x_i$ for $1 \leq i \leq |S|$. Let $\mathcal{P} = \{P_1, \ldots, P_{|S|}\}$. We add a path of

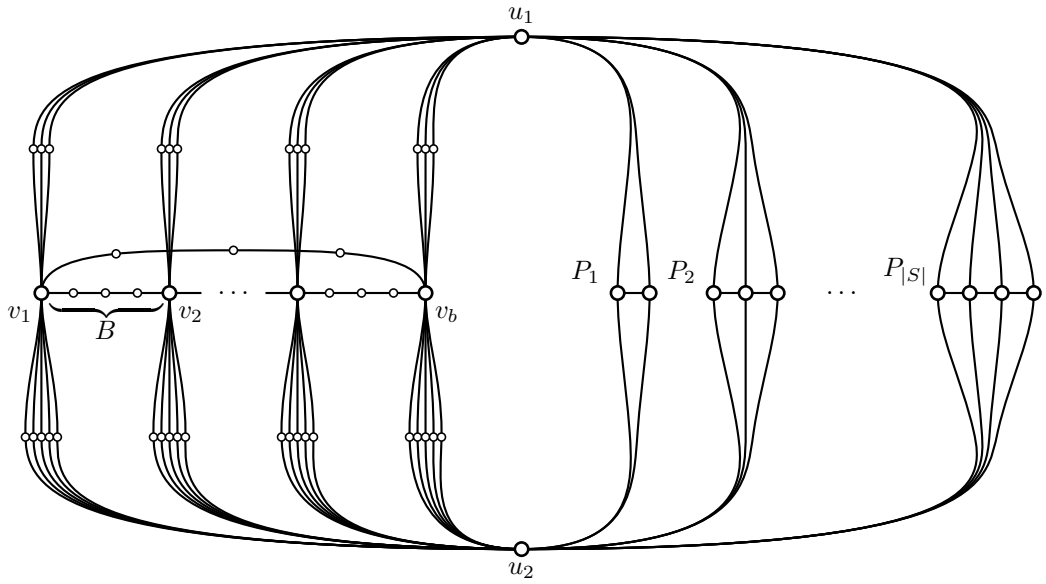


Figure 4 An illustration of the reduction for Theorem 9. Note that the number of depicted spokes between u_i and v_j is not accurate for aesthetic purposes.

length B between v_i and v_{i+1} for each $1 \leq i \leq b$, where $v_{b+1} \coloneqq v_1$. These paths form a cycle C of bB vertices, passing through v_i for all i. Next, we add an edge between u_i and each vertex in P_j for $1 \leq i \leq 2$ and $1 \leq j \leq |S|$. Let m be the number of edges in the graph constructed so far (namely, $m = 4bB - |S|$). Finally, for $1 \leq i \leq b$, we add $\ell_1 \coloneqq m+1$ spokes between u_1 and v_i and add $\ell_2 \coloneqq (\ell_1 b + m) + 1$ spokes between u_2 and v_i. See Figure 4 for an illustration of the constructed graph G. Observe that, by construction, removing $b+2$ vertices, say $u_1, u_2, v_1, \ldots, v_b$, yields a path forest. The construction can be done in polynomial time.

From a solution of Unary Bin Packing, we can obtain a 1-planar drawing of G easily, as in Figure 5. To be more precise, let $\{S_1, \ldots, S_b\}$ be a solution for Unary Bin Packing. We first draw the cycle C as a circle in such a way that the spokes between u_1 and v_i partition the interior of the circle into b regions $R_1, \ldots, R_b$, where R_i is enclosed by a spoke between u_1 and v_i, a spoke between u_1 and v_{i+1}, and a path between v_i and v_{i+1} on C. For each region R_i, we draw a path $P_j \in \mathcal{P}$ inside R_i if $x_j \in S_i$. Since the sum of the numbers of vertices of the paths drawn in R_j is exactly B, we can draw those paths so that each edge in the path between v_i and v_{i+1} is crossed exactly once. Hence, we can draw G so that each edge is crossed at most once.

For the other direction, suppose that G has a 1-planar drawing. By Corollary 3, there exists a 1-planar drawing Γ of the graph G such that all the spokes in G have no crossings. Since the spokes have no crossings in Γ, the subdrawing induced by them is planar, and hence there are two vertices v_i and v_j that belong to the outer cycle of this subdrawing. As $b \geq 3$, all the other v_ℓ's are drawn inside the region bounded by this outer cycle. Moreover, since two consecutive v_ℓ and $v_{\ell+1}$ are connected by a path in C, the two distinguished vertices v_i and v_j must be consecutive, namely $j = (i+1) \bmod b$. By appropriately renaming those vertices, we can assume that $G - \bigcup_{1 \leq i \leq |S|} V(P_i)$ are drawn as in Figure 6a. Then, there are b internally vertex-disjoint paths $Q_1, \ldots, Q_b$ between u_1 and u_2 such that Q_i is a concatenation of spokes between u_1 and v_i and between v_i and u_2 for each i. These paths separate the plane into b regions, $R_1, \ldots, R_b$ as Figure 6b. As each path P_i cannot lie in two different regions,

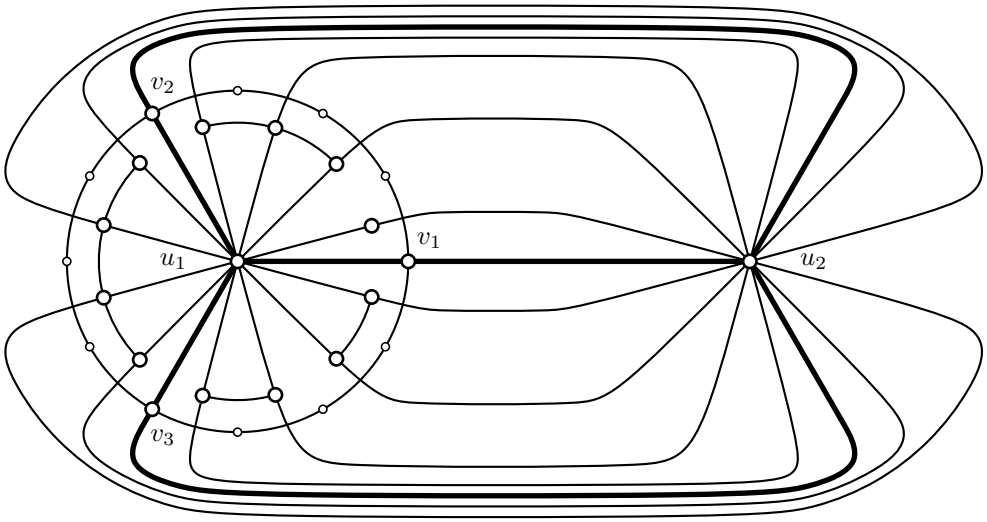
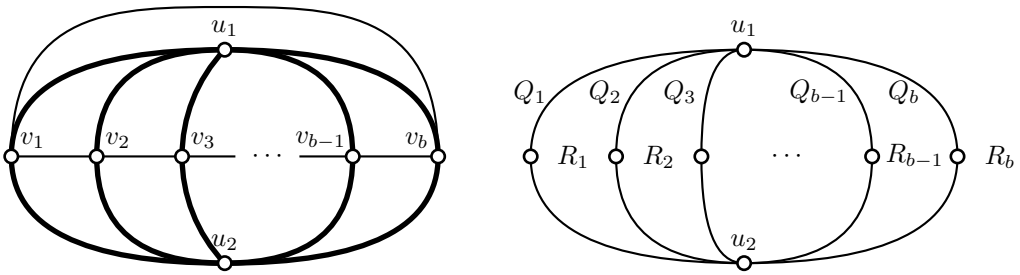


Figure 5 A 1-planar drawing of the graph constructed from an instance $\langle S = \{1, 2, 2, 3, 4\}, B = 4, b = 3\rangle$ of Unary Bin Packing. Each thick edge represents the spokes between its endpoints.



(a) **(b)**

Figure 6 The subgraphs induced by (a) cycle C and spokes (boldlines) and (b) the paths $Q_1, \ldots, Q_b$, which defines b regions $R_1, \ldots, R_b$.

$\mathcal{P}$ can be partitioned into $\{\mathcal{P}_1, \ldots, \mathcal{P}_b\}$, where $\mathcal{P}_i \subseteq \mathcal{P}$ is the collection of paths drawn inside R_i. Observe that, for each path $P_j \in \mathcal{P}_i$, there are $x_i = |V(P_i)|$ edge-disjoint paths between u_1 and u_2 that are drawn inside R_i. Each of them must cross the path between v_i and v_{i+1} on C, implying that there are at most B such paths inside R_i as Γ is 1-planar. Since there are exactly b regions, each $\mathcal{P}_i$ must satisfy $\sum_{P \in \mathcal{P}_i} |V(P)| = B$. Therefore, we can partition S into b sets such that the sum of each set is exactly B. ◀

Observe that the above reduction still works even if we add an edge between u_1 and each vertex in C, in which case the vertex set $\{u_1, u_2\}$ forms a dominating set.

▶ **Corollary 10.** *1-Planarity Testing is NP-complete even if the domination number is 2.*

As mentioned in Section 1, Cabello and Mohar [4] showed that 1-Planarity Testing is NP-hard even on near-planar graphs. Theorem 9 strengthens their hardness result as the graph G constructed in the reduction is a near-planar graph with the following properties.

▶ **Corollary 11.** *1-Planarity Testing is NP-complete on near-planar graphs with feedback vertex set number at most 3 and pathwidth at most 4.*

3.3 Twin Cover Number

In this subsection, we show that k-Planarity Testing is W[1]-hard with respect to twin cover number, which complements the positive result of the "$+k$ setting" to be presented in the subsequent section.

We give a similar reduction from Unary Bin Packing as in Section 3.2. However, in order to substitute cliques for paths, we first show the W[1]-hardness of a restricted version of Unary Bin Packing, where integers in S are sufficiently small that the constructed graph admits a B-planar drawing.

▶ **Lemma 12 (⋆).** *Unary Bin Packing is W[1]-hard parameterized by the number b of bins, even if each integer $x \in S$ satisfies $x \leq \sqrt{B} - 1$.*

Proof (sketch). We add $2bB$ copies of $B + 1$ to S of an instance of Unary Bin Packing and increase the capacity of a bin by $2B(B + 1)$ to obtain an instance with the claimed condition. We can show that there is no other solution than to distribute those $B + 1$'s evenly to b bins, making it equivalent to the original instance. ◀

▶ **Theorem 13.** *k-Planarity Testing is W[1]-hard parameterized by twin cover number.*

Proof. The reduction is almost analogous to the one used in Theorem 9. Let $I = \langle S, B, b \rangle$ be an instance of Unary Bin Packing. We can assume that $b \geq 3$ and $x \leq \sqrt{B} - 1$ for all $x \in S$ due to Lemma 12. While the basic construction of the graph G is the same as in Theorem 9, it differs in the following points:

- the cycle C contains exactly b vertices $v_1, \ldots, v_b$;
- for each $x_i \in S$, G contains a clique C_i with x_i vertices, instead of a path P_i, such that each vertex in the clique is adjacent to both u_1 and u_2;
- for each v_i, we add $\ell_1 := Bm + 1$ spokes between u_1 and v_i and add $\ell_2 := B(b\ell_1 + m) + 1$ spokes between u_2 and v_2, where m is the number of edges in G that are not contained in the spokes.

When we draw G as in Figure 5, edges incident to a vertex in clique C_i may have crossings. Each edge e in C_i may cross other edges in C_i and all the edges between u_j and a vertex in C_i for $1 \leq j \leq 2$. This implies that the number of crossings that e involves is at most

$$|E(C_i)| - 1 + 2|V(C_i)| \leq (\sqrt{B} - 1)(\sqrt{B} - 2)/2 - 1 + 2(\sqrt{B} - 1) < B.$$

Similarly, we can conclude that each edge between u_j and a vertex in C_i has at most $B - 1$ crossings. Since we can assume that all the spokes in G have no crossings due to Corollary 3, these spokes define exactly b regions, each of which contains exactly B vertices of cliques. Thus, we can show that I is a yes-instance if and only if G is B-planar as well.

As each connected component in $G - (V(C) \cup \{u_1, u_2\})$ consists of true twins, G has a twin cover of size at most $b + 2$. This completes the proof. ◀

4 FPT Algorithms

In contrast to the previous section, we mainly focus on the algorithmic aspects of k-Planarity Testing in this section and show that k-Planarity Testing is FPT when several well-known graph parameters (and k) are given as parameters.

4.1 Reducing to 1-Planarity Testing

There are several consequences from Lemma 1, combining known FPT algorithms of 1-Planarity Testing of [2].

Let G be a graph and let k be an integer. Let G_k be the graph obtained from G by subdividing each edge $k - 1$ times. Clearly, we have $\mathrm{fes}(G) = \mathrm{fes}(G_k)$. By Lemma 1, G is k-planar if and only if G_k is 1-planar. Since 1-Planarity Testing is FPT parameterized by feedback edge set number [2], the following theorem holds.

▶ **Theorem 14.** *k-Planarity Testing is FPT parameterized by feedback edge set number.*

We next observe that the treedepth of G_k is not much larger than the treedepth of G.

▶ **Lemma 15** (⋆). *For an integer $k \geq 0$, it holds that $\mathrm{td}(G_k) \leq \mathrm{td}(G) + \lceil \log_2 k \rceil$.*

Similarly to the above, we have the following theorem.

▶ **Theorem 16.** *k-Planarity Testing is FPT parameterized by treedepth $+ k$.*

We would like to note that a similar result is already mentioned in [34]. Since k-planarity is closed under vertex and edge deletions, it is known that the class of k-planar graphs with bounded treedepth is well-quasi-ordered by the induced-subgraph relation [30]. This implies a non-uniform version of Theorem 16: k-Planarity Testing can be solved by just checking whether G has an induced subgraph that belongs to a finite set $\mathcal{F}_{k,\mathrm{td}}$ of graphs, where the size of $\mathcal{F}_{k,\mathrm{td}}$ depends only on k and $\mathrm{td}(G)$.

Theorem 16 leads to FPT results for other parameters. To explain this, we introduce the notion of degeneracy. For an integer $d \geq 0$, a graph G is said to be *d-degenerate* if each nonempty subgraph of G contains a vertex of degree at most d. A graph class $\mathcal{C}$ is *degenerate* if there exists d such that every graph in $\mathcal{C}$ is d-degenerate.

▶ **Proposition 17** ([29, Proposition 6.4]). *A hereditary class of graphs $\mathcal{C}$ has bounded treedepth if and only if $\mathcal{C}$ is degenerate and does not contain a path P_t for some t as an induced subgraph.*

It is known that each k-planar graph with n vertices has at most $3.81\sqrt{kn}$ edges [1]. Since k-planarity is closed under taking a subgraph, the class of k-planar graphs is degenerate for every fixed k. Hence, combining with Proposition 17, k-Planarity Testing on P_t-free graphs parameterized by $k+t$ can be reduced to the one parameterized by treedepth $+ k$. It is known that a graph class $\mathcal{C}$ does not contain a path P_t for some t as an induced subgraph if $\mathcal{C}$ has bounded shrub-depth [14] or bounded modular-width [24].

▶ **Corollary 18.** *Let $\mathcal{C}$ be a hereditary class of graphs that excludes some path P_t as an induced subgraph. Then, k-Planarity Testing over $\mathcal{C}$ is FPT parameterized by $k+t$. In particular, k-Planarity Testing is FPT parameterized by shrub-depth $+ k$, and modular-width $+ k$.*

4.2 Polynomial Kernels

In this subsection, we describe a kernelization algorithm for k-Planarity Testing parameterized by vertex cover number $+ k$. The high-level ideas follow the polynomial kernel for 1-Planarity Testing due to Bannister, Cabello, and Eppstein [2].

Let G be a graph and let S be a vertex cover of G. We can find such a vertex cover S of size at most twice the optimum in linear time. First, we observe that the "diversity" of vertices outside of a vertex cover S is sufficiently small when G is k-planar. For a function f, let $\#_f(X)$ denote $|\{f(x) \mid x \in X\}|$, the number of possible images of X through f.

▶ **Lemma 19.** *Let $I_{\geq 2} \subseteq V(G) \setminus S$ be the set of vertices with degree at least 2 in $V(G) \setminus S$. Let π be a function that maps each vertex $v \in I_{\geq 2}$ to an unordered pair of its distinct neighbors (that is, $\pi \colon I_{\geq 2} \to \binom{N_G(v)}{2}$). If $\#_\pi(I_{\geq 2}) > |S| \cdot 3.81\sqrt{2k}$, then G is not k-planar.*

Proof. Let G_S be a graph with $V(G_S) = S$ and $E(G_S) = \pi(I_{\geq 2})$. Let G' be the graph obtained from G_S by subdividing each edge once. Then G' is a subgraph of G, and hence G' is k-planar. This implies that by Lemma 1, G_S is $2k$-planar. Hence, $\#_\pi(I_{\geq 2}) = |E(G_S)| \leq |S| \cdot 3.81\sqrt{2k}$, where the last inequality is shown in [1]. ◀

It is known that, for a positive integer k, if G contains $K_{7k+1,3}$ as a subgraph then G is not k-planar [6, 32]. Hence, the following also immediately holds.

▶ **Corollary 20.** *Let $u \in V(G) \setminus S$ be a vertex of degree at least 3. If there are at least $7k+1$ false twins of u (including u itself), then the graph G is not k-planar.*

Next, we consider vertices of degree 2. The following lemma is a key ingredient of our kernelization. It is worth mentioning that, although a similar reduction rule is used in the case $k = 1$ [2], the proof for the general case $k \geq 1$ is considerably more involved.

▶ **Lemma 21.** *Let $u \in V(G) \setminus S$ be a vertex of degree 2 with $N_G(u) = \{s_1, s_2\} \subseteq S$. If there are more than $16k^2|S|$ vertices in $V(G) \setminus S$ with the same neighborhood $\{s_1, s_2\}$, then G is k-planar if and only if $G - \{u\}$ is k-planar.*

Proof. The ($\Rightarrow$) direction is clear. Suppose that $G' \coloneqq G - \{u\}$ is k-planar and let Γ be a k-planar drawing of G'. We can assume that G' has no vertex of degree 1.

Let $T_u = \{v \in V(G') \setminus S \mid N_{G'}(v) = \{s_1, s_2\}\}$. Then, we have $t_u = |T_u| \geq 16k^2|S|$. Let $\mathcal{P}$ be the set of paths between s_1 and s_2 via a vertex in T_u. Let us align the paths in $\mathcal{P}$ according to the cyclic order in Γ around s_1, as $P_0, P_1, \ldots, P_{t_u-1}$. For two paths P_i, P_j, the *cyclic difference* of P_i and P_j is defined as $\min\{j-i, i-j+t_u\}$. We then claim the following.

▷ Claim 22. Two paths $P_i, P_j \in \mathcal{P}$ with the cyclic difference at least $4k$ do not cross in Γ.

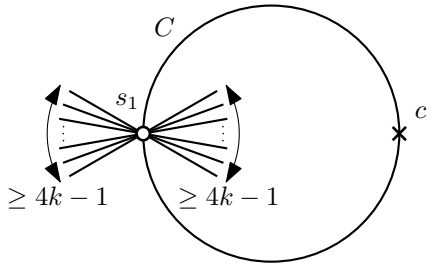


Figure 7 The Jordan curve C, the crossing point c, and the vertex s_1 with edges going out to each of the interior and exterior of C due to the cyclic difference condition.

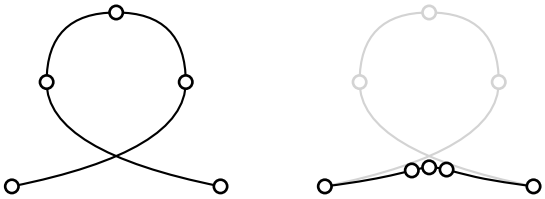


Figure 8 Let P be a path whose internal vertices have degree exactly 2. A "self-crossing" on the path can be removed by suppressing the "loop" formed by the edges in P.

Proof. Suppose that P_i and P_j do cross in Γ. Let c be a crossing point between P_i and P_j, and let C be a Jordan curve consisting of two arcs: the arc of P_i from s_1 to c and the arc of P_j from c to s_1. Such a crossing point c can be chosen so that C forms a closed curve. In both cases where s_2 is in the interior or in the exterior of regions bounded by C, from the condition that the cyclic difference of P_i and P_j is at least $4k$, there are at least $4k-1$ edges that must cross C as shown in Figure 7. As P_i, P_j can have at most $2k-1$ crossings other than c each, and hence $4k-2$ crossings in total, this leads to a contradiction. ◁

Let $\mathcal{P}' = \{P_{4ki} \mid 0 \leq i < 4k|S|\}$. As the cyclic difference of any pair of two paths in $\mathcal{P}'$ is at least $4k$, by Claim 22, no two paths in $\mathcal{P}'$ cross. We can also assume that each path in $\mathcal{P}'$ does not cross itself, as it can be resolved as shown in Figure 8. Hence, they separate the plane into $4k|S|$ regions similarly to Figure 6b. For each $0 \leq i \leq 4k|S|$, let R_i be the region that is bounded by two paths P_{4ki} and $P_{4k(i+1)}$, where $P_{4k\cdot 4k|S|} \coloneqq P_0$. Observe that, for each vertex $v \in S \setminus \{s_1, s_2\}$, the number of paths in $\mathcal{P}'$ crossed by an edge incident to $N_{G'}(v)$ is at most $4k$: when v belongs to region R_i, each edge incident to a vertex in $N_{G'}(v)$ can only be in regions at most $2k$ apart from R_i, namely, $\{R_{j \bmod 4k|S|} \mid i-2k \leq j \leq i+2k\}$. Since S is a vertex cover of G', for every edge $e \in E(G')$, $e = \{s_1, s_2\}$ (if it exists); e is contained in a path in $\mathcal{P}$; or e is incident to some vertex in $S \setminus \{s_1, s_2\}$ due to the assumption that there is no degree-1 vertex. Since at most k paths in $\mathcal{P}'$ can be crossed by $e = \{s_1, s_2\}$ (if it exists) and at most $4k(|S|-2)$ paths in $\mathcal{P}'$ can be crossed by an edge incident to a vertex in $S \setminus \{s_1, s_2\}$, there are at least $4k|S| - (4k(|S|-2)) - k = 7k$ paths in $\mathcal{P}'$ that are crossed only by the paths in $\mathcal{P}$ or not crossed at all.

Let $P \in \mathcal{P}'$ be such a path. We remove all the paths in $\mathcal{P}$ but P from Γ. Observe that now P has no crossings at all. Hence, we can redraw all the removed paths along P without making a crossing. We can also add the path (s_1, u, s_2) to Γ along P in the same way and obtain a k-planar drawing of G. ◀

By summarizing the above lemmas, we can obtain a kernelization algorithm.

▶ **Theorem 23** (⋆). *k-Planarity Testing has a kernel with $O(\mathrm{vc}(G)^2k^2\sqrt{k})$ vertices. Moreover, such a kernel can be computed in linear time.*

We can obtain a similar result for the case of neighborhood diversity.

▶ **Lemma 24** (⋆). *The vertex cover number of a k-planar graph G is at most $O(\mathrm{nd}(G)\sqrt{k})$.*

By plugging Lemma 24 to Theorem 23, we have the following.

▶ **Corollary 25.** *k-Planarity Testing has a kernel with $O(\mathrm{nd}(G)^2k^3\sqrt{k})$ vertices.*

Finally, we remark that the results of Theorem 23 and Corollary 25 cannot be generalized to vertex integrity and modular-width parameterizations (see [13, 16] for their definitions) unless NP $\subseteq$ coNP/poly, respectively. Let $G_1, \ldots, G_t$ be graphs and G be the disjoint union of $G_1, \ldots, G_t$. Let $\mathrm{vi}(G)$ be the vertex integrity of G and let $\mathrm{mw}(G)$ be the modular-width of G. Then, it holds that (1) G is 1-planar if and only if G_i is 1-planar for all $1 \leq i \leq t$ and (2) $\mathrm{vi}(G) \leq \max_i |V(G_i)|$ and $\mathrm{mw}(G) \leq \max_i |V(G_i)|$. This implies that 1-Planarity Testing parameterized by vertex integrity and by modular-width is AND-cross-compositional [3, 9, 12], implying that unless NP $\subseteq$ coNP/poly, there is no polynomial kernelization for 1-Planarity Testing of size $\mathrm{poly}(\mathrm{vi}(G))$ or $\mathrm{poly}(\mathrm{mw}(G))$. As $\mathrm{td}(G) \leq \mathrm{vi}(G)$, this kernelization lower bound also holds for treedepth parameterization.

5 Conclusion

In this paper, we study the parameterized complexity of k-Planarity Testing from the perspective of graph structural parameterizations and show several (tight) upper and lower bound results. We leave several interesting open problems relevant to our results.

When k is considered as input, we have only shown that k-Planarity Testing is FPT parameterized by feedback edge set number. It would be interesting to seek similar results using other graph parameters. In this direction, a highly related problem Crossing Number is known to be FPT parameterized by vertex cover number [19]. However, it seems that their approach cannot be directly applied to k-Planarity Testing, requiring new insights for this problem. Towards this, showing intractability with a general parameter of vertex cover number, such as vertex integrity or neighborhood diversity (see Figure 1a), would also be a nice open problem.

References

1 Eyal Ackerman. On topological graphs with at most four crossings per edge. *Comput. Geom.*, 85, 2019. doi:10.1016/J.COMGEO.2019.101574.

2 Michael J. Bannister, Sergio Cabello, and David Eppstein. Parameterized complexity of 1-planarity. *J. Graph Algorithms Appl.*, 22(1):23–49, 2018. doi:10.7155/JGAA.00457.

3 Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009. doi:10.1016/J.JCSS.2009.04.001.

4 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.

5 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

6 Július Czap and Dávid Hudák. 1-planarity of complete multipartite graphs. *Discrete Applied Mathematics*, 160(4):505–512, 2012. doi:10.1016/j.dam.2011.11.014.

7 Éric Colin de Verdière and Thomas Magnard. An FPT algorithm for the embeddability of graphs into two-dimensional simplicial complexes. In *29th Annual European Symposium on Algorithms, ESA 2021*, volume 204 of *LIPIcs*, pages 32:1–32:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.ESA.2021.32.

8 Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani. A survey on graph drawing beyond planarity. *ACM Comput. Surv.*, 52(1):4:1–4:37, 2019. doi:10.1145/3301281.

9 Andrew Drucker. New limits to classical and quantum instance compression. *SIAM J. Comput.*, 44(5):1443–1479, 2015. doi:10.1137/130927115.

10 Chandan Dubey, Uriel Feige, and Walter Unger. Hardness results for approximating the bandwidth. *Journal of Computer and System Sciences*, 77(1):62–90, 2011. Celebrating Karp's Kyoto Prize. doi:10.1016/j.jcss.2010.06.006.

11 Vida Dujmovic, Seok-Hee Hong, Michael Kaufmann, János Pach, and Henry Förster. Beyond-planar graphs: Models, structures and geometric representations (dagstuhl seminar 24062). *Dagstuhl Reports*, 14(2):71–94, 2024. doi:10.4230/DAGREP.14.2.71.

12 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019. doi:10.1017/9781107415157.

13 Jakub Gajarský, Michael Lampis, and Sebastian Ordyniak. Parameterized algorithms for modular-width. In Gregory Z. Gutin and Stefan Szeider, editors, *Parameterized and Exact Computation - 8th International Symposium, IPEC 2013, Sophia Antipolis, France, September 4-6, 2013, Revised Selected Papers*, volume 8246 of *Lecture Notes in Computer Science*, pages 163–176. Springer, 2013. doi:10.1007/978-3-319-03898-8_15.

14 Robert Ganian, Petr Hlinený, Jaroslav Nesetril, Jan Obdrzálek, and Patrice Ossona de Mendez. Shrub-depth: Capturing height of dense graphs. *Log. Methods Comput. Sci.*, 15(1), 2019. doi:10.23638/LMCS-15(1:7)2019.

15 M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic Discrete Methods*, 4(3):312–316, 1983. doi:10.1137/0604033.

16 Tatsuya Gima, Tesshu Hanaka, Masashi Kiyomi, Yasuaki Kobayashi, and Yota Otachi. Exploring the gap between treedepth and vertex cover through vertex integrity. *Theor. Comput. Sci.*, 918:60–76, 2022. doi:10.1016/J.TCS.2022.03.021.

17 Alexander Grigoriev and Hans L. Bodlaender. Algorithms for graphs embeddable with few crossings per edge. *Algorithmica*, 49(1):1–11, 2007. doi:10.1007/S00453-007-0010-X.

18 Martin Grohe. Computing crossing numbers in quadratic time. *J. Comput. Syst. Sci.*, 68(2):285–302, 2004. doi:10.1016/J.JCSS.2003.07.008.

19 Petr Hliněný and Abhisekh Sankaran. Exact crossing number parameterized by vertex cover. In *Graph Drawing and Network Visualization - 27th International Symposium, GD 2019*, volume 11904 of *Lecture Notes in Computer Science*, pages 307–319. Springer, 2019. doi:10.1007/978-3-030-35802-0_24.

20 Michael Hoffmann, Chih-Hung Liu, Meghana M. Reddy, and Csaba D. Tóth. Simple topological drawings of k-planar graphs. In David Auber and Pavel Valtr, editors, *Graph Drawing and Network Visualization - 28th International Symposium, GD 2020, Vancouver, BC, Canada, September 16-18, 2020, Revised Selected Papers*, volume 12590 of *Lecture Notes in Computer Science*, pages 390–402. Springer, 2020. doi:10.1007/978-3-030-68766-3_31.

21 Seok-Hee Hong and Takeshi Tokuyama. *Beyond Planar Graphs, Communications of NII Shonan Meetings*. Springer, 2020. doi:10.1007/978-981-15-6533-5.

22 Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *J. Comput. Syst. Sci.*, 79(1):39–49, 2013. doi:10.1016/J.JCSS.2012.04.004.

23 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, STOC 2007*, pages 382–390. ACM, 2007. doi:10.1145/1250790.1250848.

24 Minki Kim, Bernard Lidický, Tomás Masarík, and Florian Pfender. Notes on complexity of packing coloring. *Inf. Process. Lett.*, 137:6–10, 2018. doi:10.1016/J.IPL.2018.04.012.

25 Yasuaki Kobayashi, Yuto Okada, and Alexander Wolff. Recognizing 2-Layer and Outer k-Planar Graphs. In Oswin Aichholzer and Haitao Wang, editors, *Proc. 41st Annu. Sympos. Comput. Geom. (SoCG'25)*, volume 332 of *LIPIcs*, pages 65:1–65:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.SoCG.2025.65.

26 Vladimir P. Korzhik and Bojan Mohar. Minimal obstructions for 1-immersions and hardness of 1-planarity testing. *J. Graph Theory*, 72(1):30–71, 2013. doi:10.1002/JGT.21630.

27 Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Roohani Sharma, Jie Xue, and Meirav Zehavi. Crossing number in slightly superexponential time (extended abstract). In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025*, pages 1412–1424. SIAM, 2025. doi:10.1137/1.9781611978322.44.

28 Miriam Münch, Maximilian Pfister, and Ignaz Rutter. Exact and approximate k-planarity testing for maximal graphs of small pathwidth. In *Graph-Theoretic Concepts in Computer Science - 50th International Workshop, WG 2024*, volume 14760 of *Lecture Notes in Computer Science*, pages 430–443. Springer, 2024. doi:10.1007/978-3-031-75409-8_30.

29 Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-27875-4.

30 Jaroslav Nešetřil and Patrice Ossona de Mendez. On low tree-depth decompositions. *Graphs Comb.*, 31(6):1941–1963, 2015. doi:10.1007/S00373-015-1569-7.

31 János Pach and Géza Tóth. Graphs drawn with few crossings per edge. *Comb.*, 17(3):427–439, 1997. doi:10.1007/BF01215922.

32 Maximilian Pfister. *Algorithms and Combinatorics for Beyond Planar Graphs*. Phd thesis, Eberhard Karls Universität Tübingen, Tübingen, Germany, 2025. doi:10.15496/publikation-101469.

33 John C. Urschel and Jake Wellens. Testing gap k-planarity is NP-complete. *Inf. Process. Lett.*, 169:106083, 2021. doi:10.1016/J.IPL.2020.106083.

34 Meirav Zehavi. Parameterized analysis and crossing minimization problems. *Comput. Sci. Rev.*, 45:100490, 2022. doi:10.1016/J.COSREV.2022.100490.

Visualizing Treewidth

Alvin Chiu
University of California, Irvine, CA, USA

Thomas Depian
TU Wien, Austria

David Eppstein
University of California, Irvine, CA, USA

Michael T. Goodrich
University of California, Irvine, CA, USA

Martin Nöllenburg
TU Wien, Austria

Abstract

A *witness drawing* of a graph is a visualization that clearly shows a given property of a graph. We study and implement various drawing paradigms for witness drawings to clearly show that graphs have bounded pathwidth or treewidth. Our approach draws the tree decomposition or path decomposition as a tree of bags, with induced subgraphs shown in each bag, and with "tracks" for each graph vertex connecting its copies in multiple bags. Within bags, we optimize the vertex layout to avoid crossings of edges and tracks. We implement a visualization prototype for crossing minimization using dynamic programming for graphs of small width and heuristic approaches for graphs of larger width. We introduce a taxonomy of drawing styles, which render the subgraph for each bag as an arc diagram with one or two pages or as a circular layout with straight-line edges, and we render tracks either with straight lines or with orbital-radial paths.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Dynamic programming; Human-centered computing → Graph drawings

Keywords and phrases Graph drawing, witness drawings, pathwidth, treewidth

Digital Object Identifier 10.4230/LIPIcs.GD.2025.17

Related Version *Full Version*: `https://arxiv.org/abs/2508.19935` [16]

Supplementary Material *Software*: `https://doi.org/10.17605/OSF.IO/QFZ5V` [17]

Funding *Thomas Depian*: Supported by Vienna Science and Technology Fund (WWTF) [10.47379/ICT22029].
David Eppstein: Supported by NSF grant 2212129.
Michael T. Goodrich: Supported by NSF grant 2212129.
Martin Nöllenburg: Supported by Vienna Science and Technology Fund (WWTF) [10.47379/ICT22029].

1 Introduction

Work in graph drawing can be often viewed as studying paradigms for visualizing a graph, G, to clearly illustrate one or more properties of G while also optimizing one or more quality criteria, such as area or number of edge crossings; see, e.g., [19, 21, 26, 35, 53]. This viewpoint is related to the concept of a *visual proof*, which is a proof given as a visual representation called a *witness* (or visual certificate); see, e.g., [31, 46–48]. Ideally, a visual proof should be clear and concise: the property being proven should be immediately discernible simply by examining the witness. A prominent example is a crossing-free drawing as a witness for planarity. In this paper, we are interested in visualizing treewidth.


© Alvin Chiu, Thomas Depian, David Eppstein, Michael T. Goodrich, and Martin Nöllenburg;
licensed under Creative Commons License CC-BY 4.0



A *tree decomposition* of a graph, $G = (V, E)$, is a tree, T, with nodes, $V_1, V_2, \ldots, V_k$, which are called *bags*, where

1. Each V_i is a subset of V and $\bigcup_{i=1}^{k} V_i = V$.
2. If $v \in V_i$ and $v \in V_j$, then v is in each bag in the (unique) path in T from V_i to V_j.
3. For each edge $(v, w) \in E$ there is a bag that contains both v and w.

The *width* of a tree decomposition is one less than the size of its largest bag, and the *treewidth* of a graph is the smallest width of a tree decomposition. The *pathwidth* of a graph is the width of a smallest-width tree decomposition whose tree is a path. See, e.g., [13, 23, 24, 27, 37, 43] for more details regarding these topics, including applications in graph drawing and results that many NP-hard optimization problems can be solved in polynomial time for graphs with bounded treewidth or pathwidth.

We are interested in paradigms for witness drawings of graphs to certify their bounded pathwidth or treewidth. Our approaches are inspired by an example visual proof from Wikipedia that a graph has treewidth 3; see Figure 1. In this illustration, each bag is drawn as a disk with its member vertices drawn as points inside the disk. In addition, for each vertex, $v \in V$, in this drawing, there is a set of curves, each of which we call a *track*, that connect the different copies of v in the bags of the tree decomposition, i.e., the *support* of v in T. The set of tracks in T for a vertex $v \in V$ in such a drawing will form a subtree in T, which will always be a path if T is itself a path. Note, however, that even if the support for every vertex is a path, this does not imply that T is a path, e.g., as shown in Figure 1.

To provide a witness drawing for the pathwidth or treewidth of a graph, $G = (V, E)$, we add one more requirement to our visualization, which is missing from the visualization shown in Figure 1, namely *information faithfulness*, i.e., that the drawing represents the (entire) graph G and its decomposition T. In particular, we require that for each edge, $(v, w) \in E$, we must draw (v, w) as a curve inside each bag V_i such that $v, w \in V_i$ in the given tree decomposition, T, of G. Still, we are interested in such witness drawings that minimize edge and track crossings, as in Figure 1.

To see why this requirement is important, consider a graph G with pathwidth at most ω. Its *interval-width* is bounded by $\omega + 1$, i.e., G is a subgraph of an interval graph G' whose largest clique has size at most $\omega + 1$ [18]. Since interval graphs can be represented as closed intervals on the real line, such a representation of G' can be seen as a certificate that G has bounded pathwidth. However, we argue that this is not an information faithful witness drawing as it shows G' rather than G and, in particular, does not allow a reconstruction of G from the (interval) drawing of G'.

Related Work. Mehlhorn *et al.* [48] and McConnell *et al.* [46] introduce the paradigm of *certifying algorithms*, which output their result as well as a concise proof, called a "witness", that shows that the algorithm produced this output correctly. Subsequent to this work, there has been considerable work on certifying algorithms, including their inclusion in the LEDA and CGAL systems [34]. For example, additional work for witness drawings in graph drawing include work on visualizing proximity properties, such as for Delaunay graphs, Gabriel graphs, and rectangle graphs; see, e.g., [2–5, 39–42].

In GD 2024, the *GraphTrials* framework was introduced, which involves a visual proof for a graph property that can be used in an interactive proof modeled after a bench trial before a judge where a prover establishes that a graph has a given property based on its visualization [31]. Although the authors did not include pathwidth or treewidth in their GraphTrials framework, we feel that our approach to producing witness drawings for graphs with bounded pathwidth or treewidth nevertheless fits into their GraphTrials framework. In

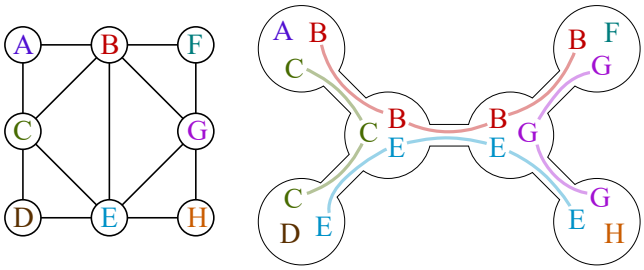


Figure 1 A witness drawing of a graph with treewidth 3. Tracks are shown color coded to illustrate the support for each vertex in the graph. Note that this visualization does not draw graph edges inside the bags of the tree decomposition. Public domain images by David Eppstein [28].

particular, in addition to faithfully representing the provided information, we argue that our created witness drawings contain all the necessary information to efficiently certify that the graph at hand has bounded pathwidth or treewidth. Moreover, such drawings can facilitate the formation of a mental model for potential judges. Thus, we believe that the drawings satisfy the required properties for visual certificates as defined in the framework [31].

We are interested in the natural optimization criterion of minimizing edge crossings in a witness drawing for a graph with bounded pathwidth or treewidth, including track-track crossings, edge-edge crossings, and track-edge crossings. In particular, few track-track crossings can aid in verifying that our decomposition satisfies Property 2, i.e., that the bags containing the vertex v induce a connected subgraph of T, for all $v \in V$. Furthermore, minimizing edge-edge and track-edge crossings improves the visual quality of the witness drawing and can thus support the verification of the (remaining) properties. Although we are not aware of any prior work on this criterion, minimizing track-track crossings is related to minimizing crossings in storyline drawings [22,33,38,54,55] and metro maps [1,6,10,11,30,50–52]. Minimization of edge-edge crossings within the bags is related to crossing minimization in linear and circular graph layouts [7,9,25,32,36,44,56].

Multiple past works include individual visualizations of tree decompositions. These include drawings of a tree with each bag shown as a set of vertices within each tree node, separate from any drawing of the graph [13,43], and drawings of a graph overlaid by bags drawn as regions that surround or connect the vertices [29]. Separately from their applications in treewidth and structural graph theory, tree decompositions with bags of non-constant size have also been used in other ways: for instance, SPQR decompositions are, essentially, tree decompositions of adhesion 2, whose bags induce 3-connected subgraphs, and these have often been visualized as a tree of 3-connected components [20,49]. However, we are unaware of past works studying visualizations of tree decompositions in any systematic way.

Our Results. In this paper, we systematize and implement witness drawings for graphs with bounded treewidth or pathwidth. Given a graph, $G = (V, E)$, our approach is to draw the vertices in each bag of a tree decomposition, T, of G as points in a disk, either in a circular layout or a linear layout. For each vertex, $v \in V$, we connect each pair of copies of v in adjacent bags in T with a curve we call a *track*, ideally with all the tracks color coded with the same color (the color for v). For each bag, V_i, in T, we draw the edges of the subgraph of G induced by V_i. In the case of a linear layout of vertices, we draw this subgraph as an arc diagram [56] (either as one-page or two-page book drawing) and in the case of a circular layout of vertices, we draw this subgraph as a circular layout with straight-line edges [9]. Our optimization goal is to minimize the total number of edge crossings, including track-track

crossings, edge-edge crossings, and track-edge crossings. See Figure 2 for three different witness drawing styles. We design and implement fixed-parameter-tractable linear-time crossing minimization algorithms for graphs with bounded pathwidth or treewidth.

Missing details for statements marked by ★ can be found in the full version [16].

2 Preliminaries

For an integer $p \geq 1$, we use $[p]$ as a shorthand for the set $\{1, 2, \ldots, p\}$. Let $G = (V, E)$ be a graph with vertex set V and edge set E. All considered graphs are simple and undirected. Throughout this paper, we use $n := |V|$ and $m := |E|$ to refer to the number of vertices and edges of G, respectively. For a tree (or path) decomposition T of G with bags $V_1, V_2, \ldots, V_k$, we let ω_T, or simply ω if T is clear from the context, denote its width. Recall $\omega_T := \max_{i \in [k]} |V_i| - 1$. We use $\overline{\omega} := \omega + 1$ as a shorthand for the maximum number of vertices in a bag. Without loss of generality, we root the tree T at an arbitrary bag. We use *decomposition* as a synonym for path and tree decompositions. Furthermore, we assume that the decomposition T consists of $k = \mathcal{O}(n)$ bags and that each bag of T has degree at most three. It is well-known that every graph of treewidth ω admits a certifying tree decomposition with $k = \mathcal{O}(n)$ fulfilling this property [14, 45]. In particular, in *nice tree decompositions*, which have found attention in the design of parameterized (dynamic programming) algorithms [18], every bag has degree at most three. With slight abuse of notation, we write $V_i \in T$ to indicate that V_i is a bag of T. Furthermore, for a bag $V_i \in T$, we let $G_i := G[V_i]$ and $E_i := E(G_i)$ denote the subgraph of G induced by V_i and its edge set, respectively.

3 A Taxonomy of Width-Witness Drawing Styles

Before we describe how to compute a witness drawing for a decomposition T of G, we examine the different witness drawing styles illustrated in Figure 2. In all of them, each bag $V_i \in T$ is represented as a disk D_i of uniform radius. We consider three different drawing styles that differ in how they arrange vertices inside each bag and how they draw the tracks of the support for each vertex. We use parenthesized letters to refer to each style.

Linear Drawings (L). In a *linear drawing*, as in Figure 2b, we arrange the vertices along a vertical line inside a disk and draw the edges as arcs to the left or right of the vertices. This drawing style is inspired by one- and two-page book drawings [36, 56]. The tracks for each support are straight-line edges connecting the respective vertices. We use L1 and L2 to differentiate between one- and two-page book drawings where needed. In an L1-drawing, all arcs, must be either to the right or to the left of the vertices, but this can change between different disks. In an L2-drawing, the arcs can be on both sides of the vertices.

Circular Drawings (C). Figure 2c shows a *circular drawing* of T. There, we arrange the vertices V_i along a circle inside the disk D_i and draw the edges E_i as straight-line chords of said circle. Combinatorially speaking, this is equivalent to a one-page book drawing of G_i [12]. As for linear drawings, we draw tracks as straight-line edges.

Orbital Drawings (O). Our last drawing style, of *orbital drawings*, is inspired by the edge routing in previous drawings with circular vertex placements [8, 15]. It is illustrated in Figure 2d and uses again a circular arrangement of the vertices. However, we now subdivide

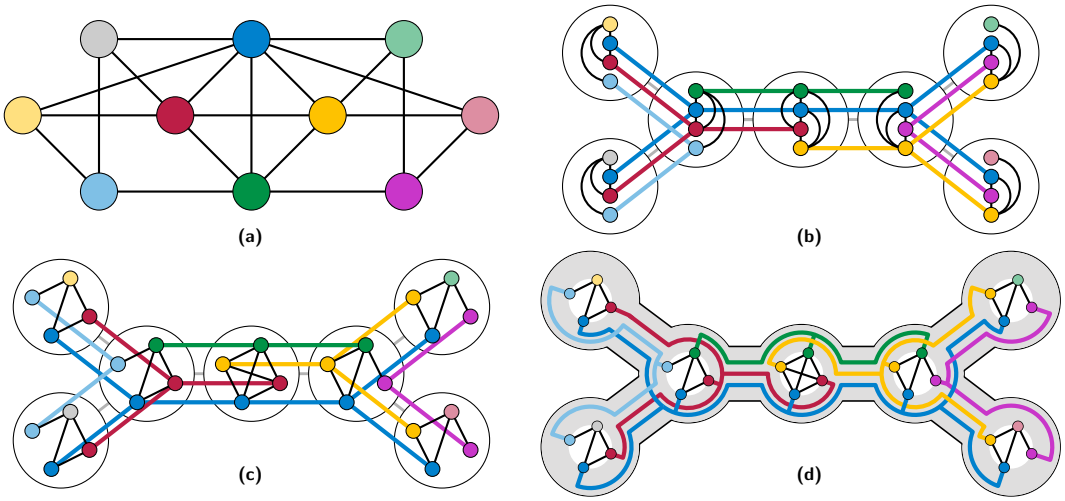


Figure 2 Witness drawings for a graph, G, with treewidth 3. A standard drawing of G is shown in **(a)**. We show in **(b)** a witness drawing of a tree decomposition, in which the subgraph for each bag is drawn as an arc diagram. The two lower witness drawings show the subgraph for each bag in a circular layout with straight-line edges. In **(c)**, the tracks connecting instances of the same vertex are drawn as straight segments, while in **(d)** the track edges are routed in orbits surrounding the bag vertices. The track edges are colored to match the vertices they represent. The drawing in **(b)** has 6 track-track crossings, no edge-edge crossings, and 6 track-edge crossings. The drawing in **(c)** has 4 track-track crossings, 1 edge-edge crossing, and 13 track-edge crossings. The drawing in **(d)** has 12 track-track crossings, 1 edge-edge crossing, and no track-edge crossings.

the disk D_i along the vertices V_i into an inner and outer part: In the inside, we draw the edges E_i as straight-line chords of the respective circle. The outside, however, the so-called *track-routing area*, is reserved for tracks. These are aligned as orbits and are not allowed to leave the track-routing area. We allow them to be routed clockwise or counterclockwise around D_i. This drawing style has no track-edge crossings and all track-track crossings occur inside the track-routing area. While this can result in a cleaner visualization, forcing tracks to stay inside the track-routing area may clutter drawings of decompositions of larger width.

In each drawing style, we evenly distribute the vertices within each disk. For linear drawings the vertex permutation uniquely defines each vertex's position inside the disk, but for the other two drawing styles this still leaves a free rotation angle. To that end, we let $0 \leq \alpha < 2\pi/\overline{\omega}$ denote the starting angle of the first vertex inside each disk, where $\alpha = 0$ corresponds to a placement at twelve o'clock (Figure 3a). We choose a single α across all disks and we select it so that no interior of a track crosses a vertex independent of the order of the vertices inside each disk. For each bag of a tree decomposition, we evenly distribute its children, if there are any, to the right of its disk (Figure 3b).

A witness drawing Γ of a given decomposition T of G consists of a drawing of (i) the tree (or path) T, (ii) each of its bags in one of the above-described drawing styles, and (iii) the support for each vertex of G. Depending on the drawing style, we call Γ an L-, C-, or O-drawing of T, or simply drawing, if the style is clear from the context. The drawing Γ might contain crossings, and we differentiate between the following three crossing types: If two tracks cross, we call it a track-track crossing (or simply t/t-crossing), if a track and an edge in a disk cross, it is a track-edge (t/e) crossing, and if two edges of G cross inside a disk, we call it an edge-edge (e/e) crossing; recall Figure 2. If neither of these crossings exist,

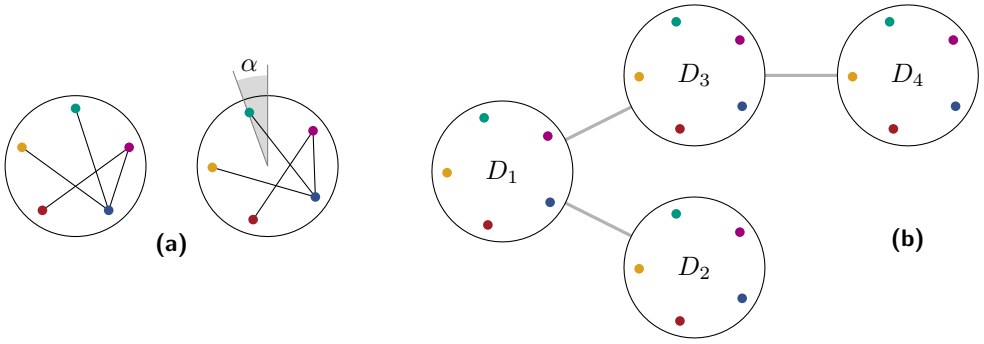


Figure 3 (a) The same circular arrangement of V_i can result in different vertex placements in the disk D_i depending on the angle α: left: $\alpha = 0$, right: $\alpha = 20°$. **(b)** Left-to-right drawing of the tree T with four bags. We omit visualizing edges and tracks to maintain readability.

we call Γ *planar*. Note that we do not count crossings with edges of T. We treat t/t-, t/e-, and e/e-crossings equally throughout the remainder of the paper. However, our algorithms can be readily adapted to weight different types of crossings differently.

4 Linear Drawings

We start our study by considering linear drawings, which visualize each G_i as a book drawing with one or two pages and draw the tracks as straight lines. Observe that the drawing of G_i is uniquely defined by the linear order $\prec_i$ of V_i and the function $\sigma_i \colon E_i \to \{\ell, r\}$, also called *page assignment*, that assigns each edge to the left or right page, respectively. In this section, we present two algorithms that compute, for a given decomposition T of constant width ω, a crossing-minimal L-drawing in linear time. Minimizing the crossings of an L-drawing is tightly interwoven with minimizing the e/e-crossings in the book drawing of each G_i, $i \in [k]$. The known hardness of the latter problem [44] allows us to show also hardness for the former problem, which justifies our focus on constant-width decompositions.

▶ **Theorem 4.1 (★).** *Deciding if a path decomposition T has an L1-drawing with at most c crossings is* NP*-hard. The same holds true for L2-, C-, and O-drawings.*

Since a path decomposition is a restricted form of a tree decomposition, Theorem 4.1 also extends to the latter type of decompositions.

4.1 Linear Drawings for Path Decompositions

We now focus on the more restrictive setting, where we want to compute a witness drawing of a path decomposition T of G of width ω. Throughout this section, we let the bags of T be ordered as they appear on the path T and assume V_1 to be the leftmost bag.

All of our algorithms, including those for C- and O-drawings, are based on the insight that, for each bag $V_i \in T$, the number of crossings involving edges and/or tracks incident to V_i is determined by the drawings of G_i in the disk D_i and the drawings of the induced subgraphs of neighboring bags. This enables us to employ a dynamic programming (DP) algorithm which processes the path decomposition from left to right and computes a crossing-minimal drawing for the first i bags, for every $i \in [k]$. Since the framework of our algorithms is identical across all drawing styles (and decomposition types), we first describe its general structure and later discuss how it can be adapted to specific drawing styles and tree decompositions.

A General DP Algorithm. We let k denote the number of bags in T and recall $\overline{\omega} = \omega + 1$. Let Γ_i be a drawing of the graph G_i in the disk D_i for each $i \in [k]$. Let $\mathcal{G}_i$ denote the set of all possible drawings Γ_i of G_i. All drawing styles admit a combinatorial representation of Γ_i, which implies that $\mathcal{G}_i$ is finite for each $i \in [k]$. Let $\Delta(\Gamma_i)$ denote a drawing of the tracks inside the disk D_i, given a drawing Γ_i of G_i, and let $\mathcal{D}(\Gamma_i)$ be the set of all such drawings. L- and C-drawings represent the tracks as straight lines. Therefore, $\Delta(\Gamma_i)$ is only relevant for O-drawings and we leave it undefined for the other two drawing styles. For this drawing style, we can again represent $\Delta(\Gamma_i)$ combinatorially, ensuring that $\mathcal{D}(\Gamma_i)$ is finite. We define $\mathcal{G}_{\max} \coloneqq \max_{i\in[k]} |\mathcal{G}_i|$ and $\mathcal{D}_{\max} \coloneqq \max_{\Gamma_i\in\mathcal{G}_i, i\in[k]} |\mathcal{D}(\Gamma_i)|$ as the maximum number of drawings of $G_1, \ldots, G_k$ and their corresponding track drawings, respectively.

In our algorithm, we maintain an $\mathcal{O}(k \cdot \mathcal{G}_{\max} \cdot \mathcal{D}_{\max})$-size table C, where $C[i, \Gamma_i, \Delta(\Gamma_i)]$ stores the minimum number of crossings of a drawing for the bags $V_1, V_2, \ldots, V_i$, where Γ_i is the drawing of G_i in D_i and $\Delta(\Gamma_i)$ the drawing of its incident tracks. We call $S_i = (i, \Gamma_i, \Delta(\Gamma_i))$ a *state* of our DP. To compute $C[S_i]$, we need to count all e/e-crossings in Γ_i involving edges from E_i, and all t/e-, and t/t-crossings involving also edges or tracks incident to vertices from V_{i-1} for a given state S_{i-1}, if the bag V_{i-1} exists. Let $\mathrm{cr}_{\mathsf{e/e}}(\cdot)$, $\mathrm{cr}_{\mathsf{t/e}}(\cdot,\cdot)$, and $\mathrm{cr}_{\mathsf{t/t}}(\cdot,\cdot)$ denote these number of crossings, respectively. Note that the exact definition of $\mathrm{cr}_{\mathsf{e/e}}(\cdot)$, $\mathrm{cr}_{\mathsf{t/e}}(\cdot,\cdot)$, and $\mathrm{cr}_{\mathsf{t/t}}(\cdot,\cdot)$ depends on the desired drawing style. For $i > 1$, the number of crossings $\mathrm{cr}(S_i, S_{i-1})$ for two states $S_i = (i, \Gamma_i, \Delta(\Gamma_i))$ and $S_{i-1} = (i-1, \Gamma_{i-1}, \Delta(\Gamma_{i-1}))$ equals the sum of $\mathrm{cr}_{\mathsf{e/e}}(S_i)$, $\mathrm{cr}_{\mathsf{t/e}}(S_i, S_{i-1})$, and $\mathrm{cr}_{\mathsf{t/t}}(S_i, S_{i-1})$. We can now relate the different states in C to each other as follows, where $S_i = (i, \Gamma_i, \Delta(\Gamma_i))$, $i \in [k]$, $\Gamma_i \in \mathcal{G}_i$, and $\Delta(\Gamma_i) \in \mathcal{D}(\Gamma_i)$:

$$C[S_i] = \begin{cases} \mathrm{cr}_{\mathsf{e/e}}(S_i) & \text{if } i = 1 \\ \min\limits_{\substack{S_{i-1} \,=\, (i-1,\Gamma_{i-1},\Delta(\Gamma_{i-1})),\\ \Gamma_{i-1}\in\mathcal{G}_{i-1},\ \Delta(\Gamma_{i-1})\in\mathcal{D}(\Gamma_{i-1})}} C[S_{i-1}] + \mathrm{cr}(S_i, S_{i-1}) & \text{otherwise} \end{cases} \tag{1}$$

Using induction, we can show that Equation (1) captures the minimum number of crossings.

▶ **Lemma 4.2 (★).** *Let T be a path decomposition of G with k bags. For each $i \in [k]$, drawings $\Gamma_i \in \mathcal{G}_i$ and $\Delta(\Gamma_i) \in \mathcal{D}(\Gamma_i)$, the table entry $C[i, \Gamma_i, \Delta(\Gamma_i)]$ equals the minimum number of crossings of a witness drawing for the bags $V_1, \ldots, V_i$ with the drawings Γ_i and $\Delta(\Gamma_i)$ for G_i and its incident tracks, respectively.*

With Lemma 4.2 at hand, we now have all key ingredients for our DP. It remains to define for each drawing style the representations of Γ_i and $\Delta(\Gamma_i)$, if needed, and to implement the crossing counting functions $\mathrm{cr}_{\mathsf{e/e}}(\cdot)$, $\mathrm{cr}_{\mathsf{t/e}}(\cdot,\cdot)$, and $\mathrm{cr}_{\mathsf{t/t}}(\cdot,\cdot)$. This will be the main task for the remainder of this and the upcoming sections.

Two-Page Linear Drawings. We now aim to compute a crossing-minimal L2-drawing of a path decomposition T. Recall that a two-page book drawing Γ_i of $G_i = (V_i, E_i)$ is uniquely defined by the linear order $\prec_i$ on V_i and the page assignment $\sigma_i \colon E_i \to \{\ell, r\}$. Thus, we set $\Gamma_i = \langle \prec_i, \sigma_i \rangle$. As the tracks are drawn using straight lines between the vertices in the respective disks, it is sufficient to store only the drawing for each G_i, $i \in [k]$, in our DP table C. As $\mathcal{G}_{\max} = \mathcal{O}(\overline{\omega}! \cdot 2^{\overline{\omega}^2})$ holds, the size of the table C is in $\mathcal{O}(k \cdot \overline{\omega}! \cdot 2^{\overline{\omega}^2})$ and it only remains to implement the crossing counting functions.

For e/e-crossings, we observe that $\mathrm{cr}_{\mathsf{e/e}}(\cdot)$ equals the number of edges with alternating endpoints in $\prec_i$ but assigned in σ_i to the same page; see Figure 4a and Equation (2).

$$\mathrm{cr}_{\mathsf{e/e}}(S_i) \coloneqq |\{(u,v),(a,b) \in E_i \mid \sigma((u,v)) = \sigma((a,b)), u \prec_i a \prec_i v \prec_i b\}| \tag{2}$$

To determine the number of t/e-crossings, it is sufficient to observe that an edge $(u, v) \in E_{i-1}$ with $u \prec_{i-1} v$ crosses with a track for the vertex $w \in V_{i-1} \cap V_i$ if and only if $u \prec_{i-1} w \prec_{i-1} v$ and $\sigma_{i-1}((u, v)) = r$ as visualized in Figure 4b. A symmetric observation can be made for edges $(u, v) \in E_i$ with $\sigma_i((u, v)) = \ell$, yielding the following function:

$$\begin{aligned} \mathrm{cr}_{\mathsf{t/e}}(S_i, S_{i-1}) \coloneqq & \sum_{\substack{(u,v)\in E_{i-1} \\ \sigma_{i-1}((u,v))=r}} |\{w \in V_{i-1} \cap V_i \mid u \prec_{i-1} w \prec_{i-1} v\}| \\ & + \sum_{\substack{(u,v)\in E_i \\ \sigma_i((u,v))=\ell}} |\{w \in V_{i-1} \cap V_i \mid u \prec_i w \prec_i v\}| \end{aligned} \tag{3}$$

This leaves us with counting the t/t-crossings, for which it is sufficient to count the number of inversions between $\prec_{i-1}$ and $\prec_i$; see Figure 4c and Equation (4).

$$\mathrm{cr}_{\mathsf{t/t}}(S_i, S_{i-1}) \coloneqq |\{u, v \in V_{i-1} \cap V_i \mid u \prec_{i-1} v, v \prec_i u\}| \tag{4}$$

Combining this with the recurrence relation from Equation (1), we obtain the following:

▶ **Theorem 4.3.** *Let T be a path decomposition of G of width ω. We can compute a crossing-minimal L2-drawing of T in $\mathcal{O}(n \cdot (\overline{\omega}!)^2 \cdot 4^{\overline{\omega}^2} \cdot \overline{\omega}^4)$ time.*

Proof. We employ our DP algorithm storing the above-described states to compute a crossing-minimal L2-drawing of T. Recall that the table C has size $\mathcal{O}(k \cdot \overline{\omega}! \cdot 2^{\overline{\omega}^2})$, where $k = \mathcal{O}(n)$ denotes the number of bags in T. To compute a single state for some $i \in [k]$, we have to access up to $\mathcal{O}(\overline{\omega}! \cdot 2^{\overline{\omega}^2})$ states for $i - 1$. A closer analysis of Equations (2)–(4) reveals that we can evaluate the function $\mathrm{cr}(\cdot, \cdot)$ in $\mathcal{O}(\overline{\omega}^4)$ time. Thus, we can compute the minimum number of crossings for a given state in $\mathcal{O}(\overline{\omega}! \cdot 2^{\overline{\omega}^2} \cdot \overline{\omega}^4)$ time. Overall, this amounts to $\mathcal{O}(k \cdot (\overline{\omega}!)^2 \cdot 4^{\overline{\omega}^2} \cdot \overline{\omega}^4)$ time to fill the entire table C. The minimum number of crossings can be obtained by taking the minimum over all $C[k, \cdot, \cdot]$. We can use standard backtracking techniques to compute the crossing-minimal L2-drawing of T. The correctness of our algorithm follows directly from the correctness of our recurrence relation, i.e., Equation (1), established in Lemma 4.2. ◀

With the use of weights, we can prioritize some types of crossings over others. Furthermore, by restricting the DP table C and relation from Equation (1), we can generalize the above-presented DP. For example, to compute an L1-drawing, we can remove the page assignment

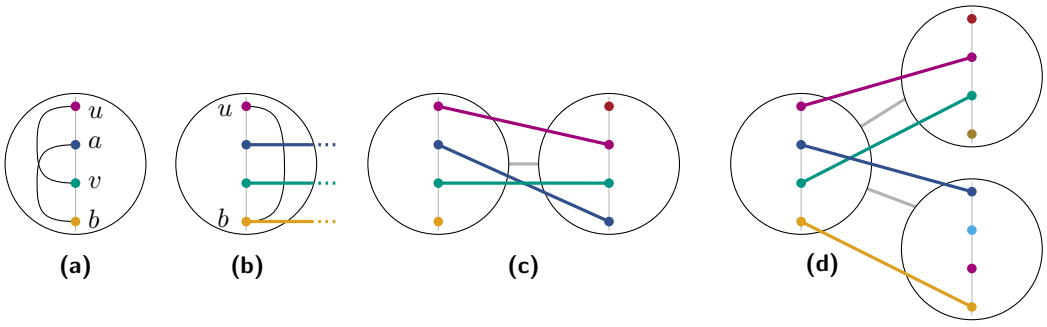


Figure 4 Illustration of different types of crossing: **(a)** The e/e-crossing between the two edges (u, v) and (a, b) is enforced by the relative order of their endpoints. **(b)** The edge (u, b) crosses the blue and green tracks because the respective vertices lie between u and b. **(c)** The blue and green tracks cause a t/t-crossing because the relative order among the respective endpoints is inversed. **(d)** A "criss-cross" t/t-crossing between blue and green that can occur in tree decompositions.

from our definition of a drawing Γ_i of G_i. Furthermore, it could be desired to enforce a consistent linear order across different bags, i.e., forbid t/t-crossings. To that end, we can consider in Equation (1) only those linear orders $\prec'$ that are consistent with $\prec$, i.e., where $\prec|_{V_{i-1}} = \prec'|_{V_i}$. Below, we summarize the effects on the running time.

▶ **Corollary 4.4.** *Let T be a path decomposition of G of width ω. We can compute a crossing-minimal*

- *L1-drawing of T in $\mathcal{O}(n \cdot (\overline{\omega}!)^2 \cdot \overline{\omega}^4)$ time.*
- *L2-drawing of T without t/t-crossings in $\mathcal{O}(n \cdot (\overline{\omega}!)^2 \cdot 4^{\overline{\omega}^2} \cdot \overline{\omega}^4)$ time.*

4.2 Linear Drawings for Tree Decompositions

Similarly, for graphs of bounded treewidth, we can obtain the minimum number of total crossings for L-drawings via DP. Given a tree decomposition T of G, we orient T "left-to-right", where the root bag is leftmost and children are to the right of the parent. We then obtain a right-to-left ordering of the tree's bags, $V_k, V_{k-1}, \ldots, V_1$, where V_1 is the root bag.

DP for Treewidth. Our DP formulation in this problem uses the same table as for path decompositions, but a given bag may now have more than one previous bag that can affect its crossings. We assume that the tree decomposition T only has bags of degree up to three. As such, any bag has at most one parent (due to the "left-to-right" orientation) and at most two children. We consider three cases to calculate $C[S_i]$ for a bag V_i, based on its in-degree $\deg^-(V_i) \in \{0, 1, 2\}$. The cases where $\deg^-(V_i)$ equal zero or one are similar to the two cases presented in the DP for pathwidth, but when $\deg^-(V_i) = 2$ the crossing function $\mathrm{cr}(\cdot,\cdot,\cdot)$ depends on the current state and its two child states. In this case, the current bag has two children belonging to its in-degree neighboring set $N^-(V_i)$ whose embedding (which child is above the other) is decided by the DP. Our DP table has the following recurrence relation:

$$C[S_i] = \begin{cases} \mathrm{cr}_{\mathsf{e}/\mathsf{e}}(S_i) & \text{if } \deg^-(V_i) = 0 \\ \min\limits_{\substack{S_x \,=\, (x, \Gamma_x, \Delta(\Gamma_x)) \\ V_x \in N^-(V_i)}} C[S_x] + \mathrm{cr}(S_i, S_x) & \text{if } \deg^-(V_i) = 1 \\ \min\limits_{\substack{(S_x, S_y) \\ V_x, V_y \in N^-(V_i), x \neq y}} C[S_x] + C[S_y] + \mathrm{cr}(S_i, S_x, S_y) & \text{if } \deg^-(V_i) = 2 \end{cases} \tag{5}$$

Two-Page Linear Drawings. The number of e/e-crossings does not change within the parent bag, so that remains the same as in Equation (2). The number of t/e-crossings can be treated independently for both children $V_x, V_y \in N^-(V_i)$, as there are no such crossings between the children. So Equation (6) is similar to Equation (3) except that we sum over both children.

$$\mathrm{cr}_{\mathsf{t}/\mathsf{e}}(S_i, S_x, S_y) := \sum_{i' \in \{x,y\}} \left(\sum_{\substack{(u,v) \in E_{i'} \\ \sigma_{i'}((u,v)) = \ell}} |\{w \in V_{i'} \cap V_i \mid u \prec_{i'} w \prec_{i'} v\}| \right.$$
$$\left. + \sum_{\substack{(u,v) \in E_i \\ \sigma_i((u,v)) = r}} |\{w \in V_{i'} \cap V_i \mid u \prec_i w \prec_i v\}| \right) \tag{6}$$

For t/t-crossings, we must handle t/t-crossings between children. Without loss of generality, let the child V_x that comes first in the input be above child V_y. Then such t/t-crossings will occur when two tracks from the children "criss-cross", where a track from the top child goes

below the track from the bottom child as in Figure 4d, captured by Equation (7). Then we add the t/t-crossings between parent and child as in Equation (4), but for both children now:

$$\mathrm{cr}_{\mathsf{t/t}}(S_i, S_x, S_y) \coloneqq |\{(u,v) \in (V_x \cap V_i) \times (V_y \cap V_i) \mid v \prec_i u\}| \tag{7}$$

$$+ \sum_{i' \in \{x,y\}} |\{u, v \in V_{i'} \cap V_i \mid u \prec_{i'} v, v \prec_i u\}| \tag{8}$$

A similar analysis to Theorem 4.3 and Corollary 4.4 yields the following results for L-drawings:

▶ **Theorem 4.5.** *Let T be a tree decomposition of G of width ω. We can compute a crossing-minimal L2-drawing of T in $\mathcal{O}(n \cdot (\overline{\omega}!)^3 \cdot 8^{\overline{\omega}^2} \cdot \overline{\omega}^4)$ time.*

Proof. The table C has the same size $\mathcal{O}(k \cdot \overline{\omega}! \cdot 2^{\overline{\omega}^2})$, with $k = \mathcal{O}(n)$, and computing $\mathrm{cr}(\cdot,\cdot,\cdot)$ still takes $\mathcal{O}(\overline{\omega}^4)$ time as in the pathwidth case. However, computing a single state must now check all possible pairings of its two child states, yielding $\mathcal{O}((\overline{\omega}! \cdot 2^{\overline{\omega}^2})^2)$ states in the worst case. So, altogether, the running time will be in $\mathcal{O}(n \cdot (\overline{\omega}!)^3 \cdot 8^{\overline{\omega}^2} \cdot \overline{\omega}^4)$. ◀

▶ **Corollary 4.6.** *Let T be a tree decomposition of G of width ω. We can compute a crossing-minimal L1-drawing of T in $\mathcal{O}(n \cdot (\overline{\omega}!)^3 \cdot \overline{\omega}^4)$.*

Finally, we note that our DP algorithm can be extended to decompositions T with bags of arbitrary degree by generalizing Equations (5)–(7). In particular, for Equation (7), we must account for t/t-crossings between every pair of children of V_i. As stated in Equation (5), we consider all possible combinations of states for V_i's children, and the number of such combinations grows with their number. Moreover, observe that the embedding of these children affects the number of t/t-crossings, and the number of potential embeddings is itself exponential in the degree d of V_i. Therefore, allowing bags with non-constant degree in T introduces an exponential factor in terms of d in the running time of our DP algorithm.

5 Drawing Styles with Circular Vertex Placements

In this section, we discuss how to adapt our DP algorithm presented in Section 4 to compute crossing-minimal C- and O-drawings.

Circular Drawings. Circular drawings arrange the vertices in every disk D_i, $i \in [k]$, on a circle and draw the edges E_i as straight lines. Thus, the drawing Γ_i of G_i is uniquely defined by the placement of the vertices V_i in D_i. Since they are evenly distributed on a circle, it suffices to store in Γ_i for each bag $V_i \in T$, the counterclockwise order $\prec_i$ of the vertices V_i as they appear in D_i, starting at the twelve o'clock position. Recall that the angle parameter α specifies the starting angle of the first vertex in the order $\prec_i$ in the disk D_i. As the tracks are again straight lines, we do not need to store $\Delta(\Gamma_i)$, reducing the size of C to $\mathcal{O}(k \cdot \overline{\omega}!)$.

When filling the table C, we recall that the recurrence relations from Equations (1) and (5) include the number of e/e-, t/e-, and t/t-crossings involving the vertices of a bag V_i, $i \in [k]$, and those of its adjacent bags (if they exist). The number of e/e-crossings corresponds, due to the equivalence with one-page book drawings, to the number of edge pairs with alternating endpoints in $\prec_i$ and is, therefore, purely combinatorial in this setting. In contrast, the t/e-, and t/t-crossings are geometric in nature and depend on the concrete positions of the vertices within the disks. Observe that these positions depend on α and we remark that the choice of α influences the number of observed crossings; see also Figure 5. Consequently, we equip the crossing counting functions $\mathrm{cr}_{\mathsf{t/e}}(\cdot,\cdot,\cdot)$ and $\mathrm{cr}_{\mathsf{t/t}}(\cdot,\cdot,\cdot)$ with the parameter α. Altogether, this is sufficient to compute crossing-minimal C-drawings.

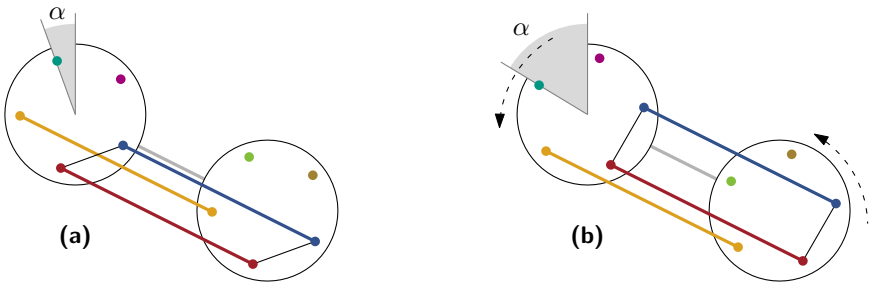


Figure 5 The observed crossings not only depend on the order of the vertices in the disks but also on the choice of α (also indicated with the dashed arrows): The t/e-crossing with the yellow track in **(a)** is not present in **(b)** although we use the same linear orders in both visualizations.

Orbital Drawings. Orbital drawings extend circular drawings by routing the tracks as orbits around the vertices; recall Figure 2d. Since tracks must remain within the track-routing area, every O-drawing is free of t/e-crossings. However, unlike in the other two drawing styles, the drawing of tracks within a disk is no longer uniquely determined by the placement of the vertices. In particular, tracks can be assigned to different orbits and may orbit the drawing Γ_i of G_i either clockwise or counterclockwise. Thus, while it still suffices to store a linear order $\prec_i$ of V_i for Γ_i, representing $\Delta(\Gamma_i)$ now requires additional care.

We model $\Delta(\Gamma_i)$ with two parts. First, to model the assignment of tracks to orbits, we introduce an *orbit assignment function* $\lambda_i \colon V_i \to [\overline{\omega}]$, where $\lambda_i(v)$ specifies the orbit used for the tracks of vertex v, numbered from the center of the disk D_i outward; see Figure 6. Two tracks for different vertices $u, v \in V_i$ may share the same orbit if they do not intersect. Otherwise, we require $\lambda_i(u) \neq \lambda_i(v)$ to guarantee that no two tracks for different vertices share the same orbit simultaneously. Second, to model the direction in which the tracks orbit, i.e., clockwise or counterclockwise, we consider the (up to two) children V_x and V_y and the parent V_q of the bag V_i in the tree decomposition T, if they exist. We define a *direction function* $\delta_i \colon V_i \times \{x, y, q\} \to \{\mathrm{cw}, \mathrm{ccw}\}$ which, for each vertex $v \in V_i$, captures the direction (clockwise (cw) or counterclockwise (ccw)) of the track from V_i to each adjacent bag. Figure 6 illustrates the semantic of δ_i and underlines the importance of storing this information separately for each adjacent bag. Thus, for a given drawing Γ_i of G_i, we set $\Delta(\Gamma_i) = (\lambda_i, \delta_i)$. This increases the size of the DP table C to $\mathcal{O}(k \cdot \overline{\omega}! \cdot (8\overline{\omega})^{\overline{\omega}})$.

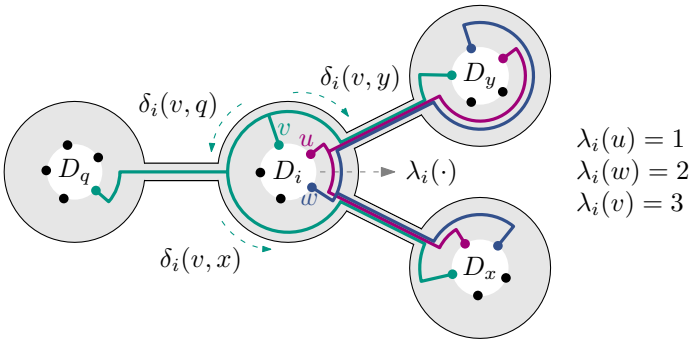


Figure 6 Visualization of a (partial) O-drawing and the information that we store in the table C. The track routing area is indicated in gray. Black vertices represent arbitrary other vertices. In this drawing, we have $\lambda_i(u) = 1$, $\lambda_i(w) = 2$, $\lambda_i(v) = 3$, $\delta_i(v, y) = \mathrm{cw}$, and $\delta_i(v, q) = \delta_i(v, x) = \mathrm{ccw}$. Note that these orientations are required to minimize the t/t-crossings involving the tracks for vertex v.

Regarding the computation of the number of crossings, we observe that the function $\mathrm{cr}_{\mathsf{e}/\mathsf{e}}(\cdot)$ from C-drawings can be reused, and that t/e-crossings do not occur in this drawing style. t/t-crossings involving a track for a vertex $v \in V_i$ occur in the following two cases. First, if v also appears in an adjacent bag, and a track for some vertex $u \in V_i$ with $\lambda_i(u) < \lambda_i(v)$ passes by the position of v in Γ_i, then these two tracks cross: The track for v must cross the one of u to reach its orbit; see Figure 7a. Second, for each adjacent bag V_j with $v \in V_j$, we count the number of vertices $u \in V_i$ such that a track for u blocks the path from D_i to D_j, see Figure 7b. In both cases, the crossings can be determined from the starting angle α, the order of the children, and the stored information, i.e., Γ_i and $\Delta(\Gamma_i)$, for the disk D_i of V_i and the disks of its adjacent bags. Hence, the function $\mathrm{cr}_{\mathsf{t}/\mathsf{t}}(\cdot,\cdot,\cdot)$ still only depends on the states for V_i and its child bags. Finally, since tracks to occurrences of the same vertex $v \in V_i$ in different bags can overlap, we must avoid double-counting in such cases; see again Figure 7a. We summarize below the findings of this section. Recall that $k = \mathcal{O}(n)$ holds.

▶ **Theorem 5.1 (★).** *Let T be a decomposition of G of width ω. We can compute a crossing-minimal*

- *C-drawing of T in $\mathcal{O}(n \cdot (\overline{\omega}!)^2 \cdot \overline{\omega}^4)$ time if T is a path decomposition.*
- *C-drawing of T in $\mathcal{O}(n \cdot (\overline{\omega}!)^3 \cdot \overline{\omega}^4)$ time if T is a tree decomposition.*
- *O-drawing of T in $\mathcal{O}(n \cdot (\overline{\omega}!)^2 \cdot (4\overline{\omega})^{2\overline{\omega}} \cdot \overline{\omega}^4)$ time if T is a path decomposition.*
- *O-drawing of T in $\mathcal{O}(n \cdot (\overline{\omega}!)^3 \cdot (8\overline{\omega})^{3\overline{\omega}} \cdot \overline{\omega}^4)$ time if T is a tree decomposition.*

6 Heuristic Approaches to Compute Linear Witness Drawings

A proof-of-concept implementation of our DP from Section 4.2 for L2-drawings of tree decompositions required over 8 minutes to terminate even for small decompositions of width $\omega = 3$ with four bags. This is no surprise given the running time bounds established in Theorem 4.5. For tree decompositions of larger width, this became too large in practice with a single state needing to check up to $(5! \cdot 2^{5^2})^2 \approx 10^{19}$ possible configurations for $\omega = 4$.

Consequently, we do not want to restrict our attention to exact, but slow, algorithms, but also explore heuristics to efficiently compute drawings with a small number of crossings, but without any formal guarantee on optimality. In this section, we describe three heuristics for L2-drawings. We evaluate their performance in Section 7 and provide in Figures 8 and 9 sample drawings computed by them.

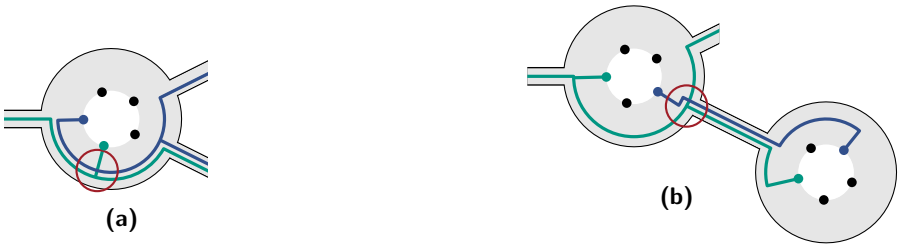


Figure 7 The two cases how t/t-crossings in O-drawings can arise: In **(a)**, the blue tracks orbit inside the green ones, thus causing a crossing when they pass the green vertex. t/t-crossings are highlighted with the red circles. The crossing in **(b)** occurs because the green track blocks the visibility to the lower-right disk for the blue track. Observe that there is only one t/t-crossing in **(a)**, although there is technically one blue track from the upper- and one from the lower-right disk.

Global Book Drawing Heuristic. Since we visualize in an L2-drawing each graph G_i, $i \in k$, as a two-page book drawing of G_i, our first heuristic computes such a drawing $\langle \prec, \sigma \rangle$ for the entire graph G. After that, we set for each disk D_i $\prec_i = \prec|_{V_i}$ and $\sigma_i(e) = \sigma(e)$ for every edge $e \in E_i$, i.e., we project the drawing $\langle \prec, \sigma \rangle$ down into each bag. If T is a tree decomposition, we additionally perform a bottom-up traversal of T, choosing for each internal bag V_i the embedding of its children that yields the fewer t/e- and t/t-crossings locally.

Since computing a crossing-minimal one-page book drawing for a graph is an NP-hard problem [44], we already employ a heuristic for this step. Klawitter, Mchedlidze, and Nöllenburg [36] performed an extensive experimentally evaluation on book drawing heuristics. In their conclusion, they recommended the heuristic `conGreedy+`, which places one vertex after the other on the spine, selecting the next vertex based on the number of already placed neighbors, and extends the spine order and page assignment greedily. While for two-page book drawings of k-trees, which are maximal graphs of treewidth k, it was slightly outperformed by other heuristics, it performed best overall and in particular for graphs with a sub-quadratic number of edges. The heuristic has a running time of $\mathcal{O}(m^2 n)$ [36] and we call it `Global conGreedy+`.

Local Book Drawing Heuristic. Computing a book drawing for the whole graph G and then applying it to each disk D_i, $i \in k$, has two main disadvantages. On the one hand, we usually do not visualize the entire graph G in a single bag, but several induced subgraphs G_i separately. Therefore, the drawing $\langle \prec_i, \sigma_i \rangle$ for G_i might contain unnecessary crossings. On the other hand, the above-described heuristic is, apart from the embedding step, unaware of t/e- and t/t-crossings. This, in particular, leads to situations where swapping the page assignment of one edge $e \in E_i$ or the entire edge set E_i can dramatically reduce the number of t/e-crossings. The latter occurs especially at the root and leave bags of T.

Therefore, we now employ the algorithm `conGreedy+` in each bag V_i separately during a top-down traversal of T, thus computing for each G_i a book drawing from scratch. Since the drawing for the parent of a bag V_i has been determined, we can take the potential t/t- and t/e-crossings into account. More precisely, when selecting the spine position of a vertex $v \in V_i$ and the page assignment for its incident edges in G_i we also compute the number of t/e- and t/t-crossings that this causes based on the drawing of the parent, similar to Equations (6) and (7). Furthermore, we forward this information to its children when considering their two possible embeddings if necessary. We call this heuristic `Local conGreedy+` and, since the size of each G_i is bounded, its running time is $\mathcal{O}(n \cdot \overline{\omega}^5)$.

Local Search. In addition to the two above-mentioned heuristics, we also perform a local search to reduce the number of crossings. Given an L2-drawing of T, we traverse T bottom up. For each bag $V_i \in T$, we perform one or several of the following movements.

Vertex-Swap: Take two vertices $u, v \in V_i$ and swap their position on the spine $\prec_i$.
Edge-Swap: Take two edges $e_1, e_2 \in E_i$ with $\sigma_i(e_1) \neq \sigma_i(e_2)$ and swap their page assignment.
Edge-Flip: Take an edge $e \in E_i$ and assign it to the other page.
Embedding-Flip: Flip the embedding of the subtree V_i, i.e., the order of its children.

For a given bag $V_i \in T$, we perform a hill-climbing approach and exhaustively apply above movements until no further improvement can be made in the drawing $\langle \prec_i, \sigma_i \rangle$. Then, we proceed to the next bag $V_j \in T$. Note that when evaluating the number of crossings, we keep the drawings in the remaining bags fixed. Thus, after a bottom-up traversal of T, we perform a second traversal, this time top-down. Afterwards, or after a predetermined period of time, we return the best solution found so far.

7 Experimental Evaluation

We implemented the dynamic program for L2-crossings as described in Section 4.2 as well as all of the heuristics presented in Section 6 in `Python 3.6.9`. All algorithms run on a system with Intel Xeon E5-2640 v4 10-core processors at 2.40 GHz. We set hard limits for memory and time of 96 GB and fifteen minutes, respectively, but memory was not a limitation for our algorithms. If we improve a solution from a heuristic using local search, we indicate this using "`& LS`". Also the local search has a maximum running time of fifteen minutes. The computed drawings are visualized using `d3.js` on an independent system and this final step is not considered here. The instances are based on a public repository containing graphs extracted from "Sage" together with a tree decomposition of them.[1] Out of the 150 graphs, 55 of them were equipped with a (tree or path) decomposition T where each bag, i.e., node of the decomposition T, has degree at most three in T and we used these in our experiments. We have no information on how these decompositions were computed or if they are optimal. The width of the decompositions ranges between one and 53 and they have between one and 143 bags. For 60% of the instances, both parameters were simultaneously below ten. The graphs have between four and 143 vertices and between six and 1,008 edges, with a mean density, i.e., $2m/(n(n-1))$, of 0.37.

Computed Drawings. We provide in Figures 8 and 9 sample drawings of two graphs together with the witness drawings computed by our different algorithms. Generally, we can see that the DP and `Global conGreedy+` produce drawings with a more consistent arrangement of the vertices on the individual spines. Comparing Figures 8b and f, we can observe that the obtained drawing from `Global conGreedy+ & LS` is nearly identical to the one of the DP, confirming the observed low crossing numbers. The consistency across different spines yields also a visually more pleasing drawing, as we can see in Figure 9. We provide in the full version [16] more sample drawings computed by our algorithms and analyze below their running time and the quality of the produced drawings.

Running Times, Tradeoffs, and the Influence of the Width. All heuristics terminated on every instance within fifteen minutes. The DP could solve only thirteen instances within the time limit. To evaluate the crossing minimization performance of the heuristics on a larger set of instances, we ran the DP with a time limit of six hours, after which 21 of the 55 instances, all of width at most four, could be solved optimally. For the remaining 34 instances, `Global conGreedy+` obtained four, `Local conGreedy+` three, `Global conGreedy+ & LS` 18, and `Local conGreedy+ & LS` 20 times a solution with the fewest crossings.

Figure 10 shows the relative number of crossings, compared to the best-performing algorithm, i.e., the DP where available and one of the heuristics otherwise, and running time for each algorithm and instance. We group instances by width and sort them within each group based on the number of bags. In the full version [16], we provide a filtered version of Figure 10, focusing on instances for which the DP terminated within six hours. Regarding the running time, we can observe in Figure 10a that all heuristics without the local search step terminated within a second on every instance except the largest ones. As

[1] The original dataset is available at `https://github.com/freetdi/named-graphs`. The filtered dataset, source code, and evaluation code can be found on OSF [17]. The application can also be accessed online at `https://www.ac.tuwien.ac.at/projects/visualizing-treewidth/`.

[2] A drawing of the graph can be found online: `https://en.wikipedia.org/wiki/Brinkmann_graph`.

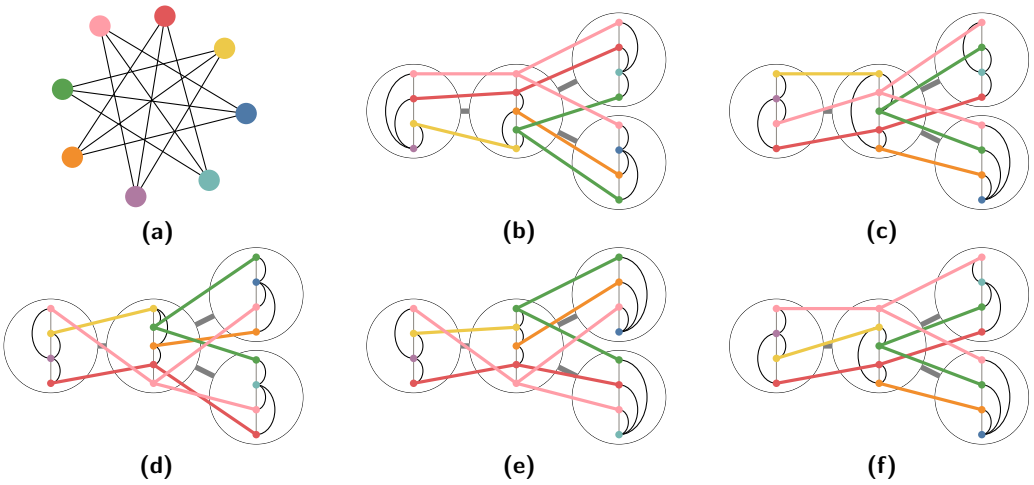


Figure 8 Comparison of the drawings computed by our algorithms for the decomposition of the *Wagner* graph; we indicate the number of crossings in the parenthesis. **(a)** Force-based drawing of the graph, **(b)** DP (3), **(c)** `Global conGreedy+` (8), **(d)** `Local conGreedy+` (9), **(e)** `Local conGreedy+ & LS` (5), and **(f)** `Global conGreedy+ & LS` (4).

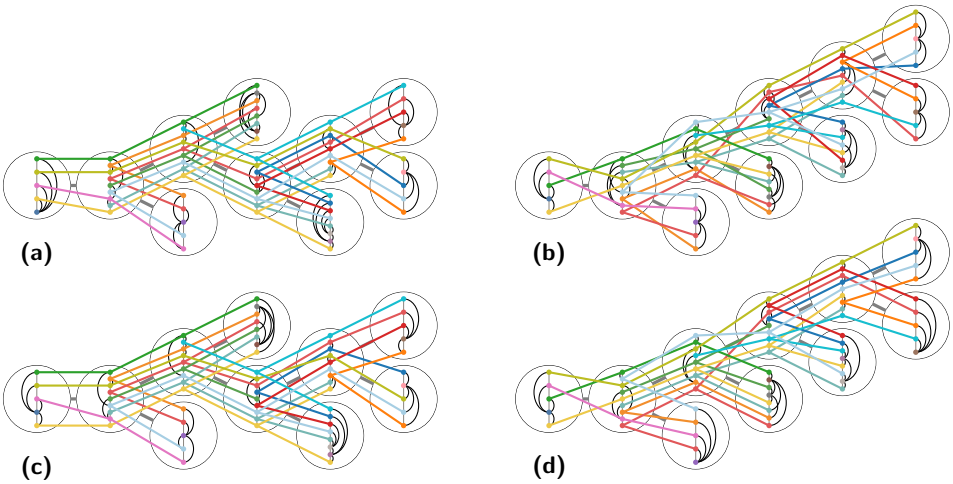


Figure 9 Comparison of the drawings computed by our heuristics for the *Brinkmann* graph[2]; we indicate the number of crossings in the parenthesis. **(a)** `Global conGreedy+` (113), **(b)** `Local conGreedy+` (105), **(c)** `Global conGreedy+ & LS` (64), and **(d)** `Local conGreedy+ & LS` (72).

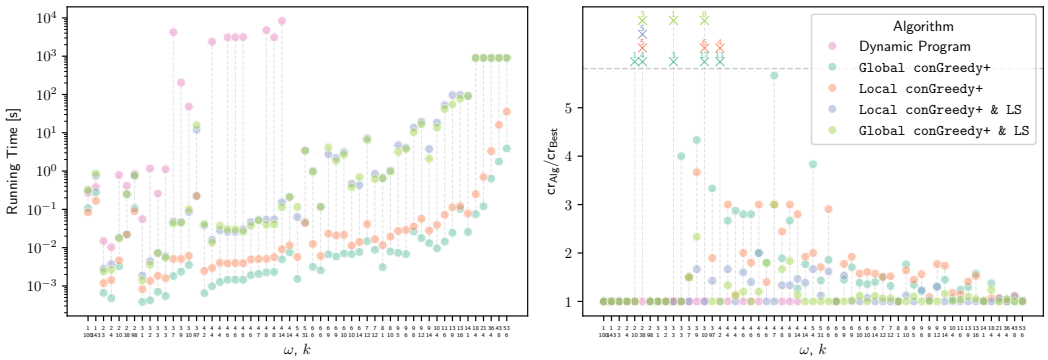


Figure 10 (a) Wall-clock running time ($\log_{10}$-scale) of the heuristics and **(b)** their optimality-ratio compared to the best found solution for different values of ω and number of bags k. We group instances based on ω, sort each group by k, and indicate on the x-axis the value of ω and k in the first and second row, respectively. Vertical lines connect data points for one instance.

expected, `Global conGreedy+` is the fasted heuristic, since it has to compute only one book drawing for the entire graph. Regarding the optimality-ratio, Figure 10b shows that for instances with larger width, `Global conGreedy+` seems to work slightly better than `Local conGreedy+`. We believe that computing a book drawing for each G_i individually could yield situations where the obtained spine orders yield many `t`/`t`-crossings. In particular, we suspect that spine orders that "locally" admit few crossings could "globally" introduce, for example, unexpected "criss-cross" `t`/`t`-crossings (recall Figure 4d). We see the strength of our DP for decompositions of width two or three. For larger width, the high running time becomes impractical. However, small-width instances are often simple enough that all heuristics can solve them to optimality in a few milliseconds. Interestingly, for instances with very large treewidth, the benefit of local search drastically decreases: Despite performing movement operations for the entire fifteen minutes, the obtained solution is, if at all, only negligibly better then the corresponding heuristic without the local search. However, for instances of moderate width, local search considerably improved the initial solution.

8 Concluding Remarks and Future Directions

In this paper, we have initiated the study of visualizing tree and path decompositions of graphs to visually certify their treewidth and pathwidth, respectively. Our experimental study shows that for decompositions of width at most three, we can compute exact crossing-minimal drawings in a few seconds. For decompositions of medium to larger width, the heuristics combined with a local search step seem to be the preferred method. We see the computation of a crossing-minimal drawing where we do not restrict the placement of the child bags or the starting angle inside the disk as a challenging open algorithmic problem. Furthermore, the taxonomy presented Section 3 can be seen as a first step towards exploring the design space for witness drawings of graphs with bounded pathwidth or treewidth. In particular, future work should explore further drawing styles for (i) the graph G_i inside each bag V_i, (ii) tracks between the bags, (iii) the underlying tree (or path) T, and possibilities (iv) to distinguish different vertices other than color. On top of that, a comprehensive design study can also consider (v) the option to not visualize an edge uv in every bag V_i with $u, v \in V_i$ but only

in one such bag, (vi) the algorithmic and visual impact of the structure of the underlying tree/path T, e.g., its number of bags, their maximum degree, or the "niceness" of T, and (vii) further optimization criteria other than the (unweighted) number of crossings.

Additionally, we believe that the exploration of the design space should go hand in hand with the development of further exact and heuristic algorithms and an empirical evaluation of the drawing styles proposed in this article or extracted from the (extended) taxonomy, e.g., with a user study. Finally, designing and computing suitable witness drawings for other width parameters of graphs is an interesting, independent direction for future work.

References

1 Evmorfia N. Argyriou, Michael A. Bekos, Michael Kaufmann, and Antonios Symvonis. On metro-line crossing minimization. *Journal of Graph Algorithms and Applications*, 14(1):75–96, 2010. doi:10.7155/jgaa.00199.

2 Boris Aronov, Muriel Dulieu, and Ferran Hurtado. Witness (delaunay) graphs. *Computational Geometry*, 44(6):329–344, 2011. doi:10.1016/j.comgeo.2011.01.001.

3 Boris Aronov, Muriel Dulieu, and Ferran Hurtado. Witness rectangle graphs. In Frank Dehne, John Iacono, and Jörg-Rüdiger Sack, editors, *Proc 12th International Symposium on Algorithms and Data Structures (WADS'11)*, volume 6844 of *Lecture Notes in Computer Science*, pages 73–85. Springer, 2011. doi:10.1007/978-3-642-22300-6_7.

4 Boris Aronov, Muriel Dulieu, and Ferran Hurtado. Witness gabriel graphs. *Computational Geometry*, 46(7):894–908, 2013. doi:10.1016/j.comgeo.2011.06.004.

5 Boris Aronov, Muriel Dulieu, and Ferran Hurtado. Witness rectangle graphs. *Graphs and Combinatorics*, 30(4):827–846, 2014. doi:10.1007/s00373-013-1316-x.

6 Matthew Asquith, Joachim Gudmundsson, and Damian Merrick. An ILP for the metro-line crossing problem. In James Harland and Prabhu Manyem, editors, *Proc. 14th Computing: The Australasian Theory Symposium (CATS'08)*, volume 77 of *CRPIT*, pages 49–56. Australian Computer Society, 2008. URL: http://crpit.scem.westernsydney.edu.au/abstracts/CRPITV77Asquith.html.

7 Michael J. Bannister and David Eppstein. Crossing minimization for 1-page and 2-page drawings of graphs with bounded treewidth. In Christian Duncan and Antonios Symvonis, editors, *Proc. 22nd International Symposium on Graph Drawing and Network Visualization (GD'14)*, volume 8871 of *Lecture Notes in Computer Science*, pages 210–221. Springer, 2014. doi:10.1007/978-3-662-45803-7_18.

8 Lukas Barth, Benjamin Niedermann, Ignaz Rutter, and Matthias Wolf. A topology-shape-metrics framework for ortho-radial graph drawing. *Discrete & Computational Geometry*, 70(4):1292–1355, 2023. doi:10.1007/s00454-023-00593-y.

9 Michael Baur and Ulrik Brandes. Crossing reduction in circular layouts. In Juraj Hromkovič, Manfred Nagl, and Bernhard Westfechtel, editors, *Proc. 30th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'04)*, volume 3353 of *LNCS*, pages 332–343. Springer Berlin Heidelberg, 2004. doi:10.1007/978-3-540-30559-0_28.

10 Michael A. Bekos, Michael Kaufmann, Katerina Potika, and Antonios Symvonis. Line crossing minimization on metro maps. In Seok-Hee Hong, Takao Nishizeki, and Wu Quan, editors, *Proc. 15th International Symposium on Graph Drawing and Network Visualization (GD'07)*, volume 4875 of *Lecture Notes in Computer Science*, pages 231–242. Springer, 2007. doi:10.1007/978-3-540-77537-9_24.

11 Marc Benkert, Martin Nöllenburg, Takeaki Uno, and Alexander Wolff. Minimizing intra-edge crossings in wiring diagrams and public transportation maps. In Michael Kaufmann and Dorothea Wagner, editors, *Proc. 14th International Symposium on Graph Drawing and Network Visualization (GD'06)*, volume 4372 of *Lecture Notes in Computer Science*, pages 270–281. Springer, 2006. doi:10.1007/978-3-540-70904-6_27.

12 Frank Bernhart and Paul C. Kainen. The book thickness of a graph. *Journal of Combinatorial Theory, Series B*, 27(3):320–331, 1979. doi:10.1016/0095-8956(79)90021-2.

13 Hans L. Bodlaender. Treewidth: Characterizations, applications, and computations. In Fedor V. Fomin, editor, *Proc. 32nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG'06)*, volume 4271 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006. doi:10.1007/11917496_1.

14 Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21(2):358–402, 1996. doi:10.1006/jagm.1996.0049.

15 Annika Bonerath, Martin Nöllenburg, Soeren Terziadis, Markus Wallinger, and Jules Wulms. Boundary labeling in a circular orbit. In Stefan Felsner and Karsten Klein, editors, *Proc. 32nd International Symposium on Graph Drawing and Network Visualization (GD'24)*, volume 320 of *LIPIcs*, pages 22:1–22:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.GD.2024.22.

16 Alvin Chiu, Thomas Depian, David Eppstein, Michael T. Goodrich, and Martin Nöllenburg. Visualizing treewidth, 2025. doi:10.48550/arXiv.2508.19935.

17 Alvin Chiu, Thomas Depian, David Eppstein, Michael T. Goodrich, and Martin Nöllenburg. Visualizing treewidth: Supplementary material, 2025. Source Code; Last accessed: 2025-08-18. doi:10.17605/OSF.IO/QFZ5V.

18 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

19 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

20 Giuseppe Di Battista and Roberto Tamassia. On-line maintenance of triconnected components with SPQR-trees. *Algorithmica*, 15(4):302–318, 1996. doi:10.1007/s004539900017.

21 Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani. A survey on graph drawing beyond planarity. *ACM Computing Surveys*, 52(1):4:1–4:37, 2019. doi:10.1145/3301281.

22 Alexander Dobler, Michael Jünger, Paul J. Jünger, Julian Meffert, Petra Mutzel, and Martin Nöllenburg. Revisiting ILP models for exact crossing minimization in storyline drawings. In Stefan Felsner and Karsten Klein, editors, *Proc. 32nd International Symposium on Graph Drawing and Network Visualization (GD'24)*, volume 320 of *LIPIcs*, pages 31:1–31:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.GD.2024.31.

23 Vida Dujmović, David Eppstein, and David R. Wood. Structure of graphs with locally restricted crossings. *SIAM Journal on Discrete Mathematics*, 31(2):805–824, 2017. doi:10.1137/16M1062879.

24 Vida Dujmović, Pat Morin, and David R. Wood. Layout of graphs with bounded tree-width. *SIAM Journal on Computing*, 34(3):553–579, 2005. doi:10.1137/S0097539702416141.

25 Vida Dujmović and David R. Wood. On linear layouts of graphs. *Discrete Mathematics & Theoretical Computer Science*, 6(2):339–358, 2004. doi:10.46298/DMTCS.317.

26 John Ellson, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Gordon Woodhull. Graphviz - open source graph drawing tools. In Petra Mutzel, Michael Jünger, and Sebastian Leipert, editors, *Proc. 9th International Symposium on Graph Drawing and Network Visualization (GD'01)*, volume 2265 of *Lecture Notes in Computer Science*, pages 483–484. Springer, 2001. doi:10.1007/3-540-45848-4_57.

27 David Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27(3):275–291, 2000. doi:10.1007/s004530010020.

28 David Eppstein. Tree decomposition.svg via Wikimedia Commons, 2010. Last accessed: 2025-08-14. URL: https://upload.wikimedia.org/wikipedia/commons/a/a7/Tree_decomposition.svg.

29 David Eppstein. What is ... treewidth? *Notices of the AMS*, 72(2):172–175, 2025. doi:10.1090/noti3043.

30 Martin Fink and Sergey Pupyrev. Metro-line crossing minimization: Hardness, approximations, and tractable cases. In Stephen Wismath and Alexander Wolff, editors, *Proc. 21st International Symposium on Graph Drawing and Network Visualization (GD'13)*, volume 8242 of *Lecture Notes in Computer Science*, pages 328–339. Springer, 2013. doi:10.1007/978-3-319-03841-4_29.

31 Henry Förster, Felix Klesen, Tim Dwyer, Peter Eades, Seok-Hee Hong, Stephen G. Kobourov, Giuseppe Liotta, Kazuo Misue, Fabrizio Montecchiani, Alexander Pastukhov, and Falk Schreiber. GraphTrials: Visual proofs of graph properties. In Stefan Felsner and Karsten Klein, editors, *Proc. 32nd International Symposium on Graph Drawing and Network Visualization (GD'24)*, volume 320 of *LIPIcs*, pages 16:1–16:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.GD.2024.16.

32 Emden R. Gansner and Yehuda Koren. Improved circular layouts. In Michael Kaufmann and Dorothea Wagner, editors, *Proc. 14th International Symposium on Graph Drawing and Network Visualization (GD'06)*, volume 4372 of *Lecture Notes in Computer Science*, pages 386–398. Springer, 2006. doi:10.1007/978-3-540-70904-6_37.

33 Martin Gronemann, Michael Jünger, Frauke Liers, and Francesco Mambelli. Crossing minimization in storyline visualization. In Yifan Hu and Martin Nöllenburg, editors, *Proc. 24th International Symposium on Graph Drawing and Network Visualization (GD'16)*, volume 9801 of *Lecture Notes in Computer Science*, pages 367–381. Springer, 2016. doi:10.1007/978-3-319-50106-2_29.

34 Michael Hoffmann, Lutz Kettner, and Stefan Näher. Two computational geometry libraries: LEDA and CGAL. In *Handbook of Discrete and Computational Geometry*, pages 1799–1831. Chapman and Hall/CRC, 2017.

35 Michael Jünger and Petra Mutzel. *Graph Drawing Software*. Springer, 2012.

36 Jonathan Klawitter, Tamara Mchedlidze, and Martin Nöllenburg. Experimental evaluation of book drawing algorithms. In Fabrizio Frati and Kwan-Liu Ma, editors, *Proc. 25th International Symposium on Graph Drawing and Network Visualization (GD'17)*, volume 10692 of *Lecture Notes in Computer Science*, pages 224–238. Springer, 2017. doi:10.1007/978-3-319-73915-1_19.

37 Ephraim Korach and Nir Solel. Tree-width, path-widt, and cutwidth. *Discrete Applied Mathematics*, 43(1):97–101, 1993. doi:10.1016/0166-218X(93)90171-J.

38 Irina Kostitsyna, Martin Nöllenburg, Valentin Polishchuk, André Schulz, and Darren Strash. On minimizing crossings in storyline visualizations. In Emilio Di Giacomo and Anna Lubiw, editors, *Proc. 23rd International Symposium on Graph Drawing and Network Visualization (GD'15)*, volume 9411 of *Lecture Notes in Computer Science*, pages 192–198. Springer, 2015. doi:10.1007/978-3-319-27261-0_16.

39 Dieter Kratsch, Ross M. McConnell, Kurt Mehlhorn, and Jeremy P. Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. In *Proc. 14th ACM-SIAM Symposium on Discrete Algorithms (SODA'03)*, pages 158–167. ACM/SIAM, 2003. URL: http://dl.acm.org/citation.cfm?id=644108.644137.

40 Dieter Kratsch, Ross M. McConnell, Kurt Mehlhorn, and Jeremy P. Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. *SIAM Journal on Computing*, 36(2):326–353, 2006. doi:10.1137/S0097539703437855.

41 William J. Lenhart and Giuseppe Liotta. Mutual witness Gabriel drawings of complete bipartite graphs. *Theoretical Computer Science*, 974:114115, 2023. doi:10.1016/j.tcs.2023.114115.

42 William J. Lenhart and Giuseppe Liotta. Mutual witness gabriel drawings of complete bipartite graphs. In Patrizio Angelini and Reinhard von Hanxleden, editors, *Proc. 30th Proc. 21st International Symposium on Graph Drawing and Network Visualization (GD'22)*, volume 13764 of *Lecture Notes in Computer Science*, pages 25–39. Springer, 2023. doi:10.1007/978-3-031-22203-0_3.

43 Silviu Maniu, Pierre Senellart, and Suraj Jog. An experimental study of the treewidth of real-world graph data. In Pablo Barceló and Marco Calautti, editors, *Proc. 22nd International*

Conference on Database Theory (ICDT'19), volume 127 of *LIPIcs*, pages 12:1–12:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.ICDT.2019.12.

44 Sumio Masuda, Toshinobu Kashiwabara, Kazuo Nakajima, and Toshio Fujisawa. On the NP-completeness of a computer network layout problem. In *Proc. IEEE Intl. Symposium on Circuits and Systems*, pages 292–295, 1987.

45 Jiři Matoušek and Robin Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Mathematics*, 108(1–3):343–364, 1992. doi:10.1016/0012-365x(92)90687-b.

46 Ross M. McConnell, Kurt Mehlhorn, Stefan Näher, and Pascal Schweitzer. Certifying algorithms. *Computer Science Review*, 5(2):119–161, 2011. doi:10.1016/j.cosrev.2010.09.009.

47 Kurt Mehlhorn, Stefan Näher, Thomas Schilz, Stefan Schirra, Michael Seel, Raimund Seidel, and Christian Uhrig. Checking geometric programs or verification of geometric structures. In Sue Whitesides, editor, *Proc. 12th International Symposium on Computational Geometry (SoCG'96)*, SCG '96, pages 159–165. ACM, 1996. doi:10.1145/237218.237344.

48 Kurt Mehlhorn, Stefan Näher, Michael Seel, Raimund Seidel, Thomas Schilz, Stefan Schirra, and Christian Uhrig. Checking geometric programs or verification of geometric structures. *Computational Geometry*, 12(1-2):85–103, 1999. doi:10.1016/S0925-7721(98)00036-4.

49 Petra Mutzel. The SPQR-tree data structure in graph drawing. In Jos C. M. Baeten, Jan Karel Lenstra, Joachim Parrow, and Gerhard J. Woeginger, editors, *Proc. 30th International Colloquium on Automata, Languages and Programming (ICALP'03)*, volume 2719 of *Lecture Notes in Computer Science*, pages 34–46. Springer, 2003. doi:10.1007/3-540-45061-0_4.

50 Martin Nöllenburg. An improved algorithm for the metro-line crossing minimization problem. In David Eppstein and Emden R. Gansner, editors, *Proc. 17th International Symposium on Graph Drawing and Network Visualization (GD'09)*, volume 5849 of *Lecture Notes in Computer Science*, pages 381–392. Springer, 2009. doi:10.1007/978-3-642-11805-0_36.

51 Sergey Pupyrev, Lev Nachmanson, Sergey Bereg, and Alexander E. Holroyd. Edge routing with ordered bundles. In Marc J. van Kreveld and Bettina Speckmann, editors, *Proc. 19th International Symposium on Graph Drawing and Network Visualization (GD'11)*, volume 7034 of *Lecture Notes in Computer Science*, pages 136–147. Springer, 2012. doi:10.1007/978-3-642-25878-7_14.

52 Sergey Pupyrev, Lev Nachmanson, Sergey Bereg, and Alexander E. Holroyd. Edge routing with ordered bundles. *Computational Geometry*, 52:18–33, 2016. doi:10.1016/j.comgeo.2015.10.005.

53 Roberto Tamassia, editor. *Handbook of Graph Drawing and Visualization*. Chapman and Hall/CRC, 2013.

54 Thomas C. van Dijk, Martin Fink, Norbert Fischer, Fabian Lipp, Peter Markfelder, Alexander Ravsky, Subhash Suri, and Alexander Wolff. Block crossings in storyline visualizations. *Journal of Graph Algorithms and Applications*, 21(5):873–913, 2017. doi:10.7155/jgaa.00443.

55 Thomas C. van Dijk, Fabian Lipp, Peter Markfelder, and Alexander Wolff. Computing storyline visualizations with few block crossings. In Fabrizio Frati and Kwan-Liu Ma, editors, *Proc. 25th International Symposium on Graph Drawing and Network Visualization (GD'17)*, volume 10692 of *Lecture Notes in Computer Science*, pages 365–378. Springer, 2017. doi:10.1007/978-3-319-73915-1_29.

56 Martin Wattenberg. Arc diagrams: Visualizing structure in strings. In Pak Chung Wong and Keith Andrews, editors, *Proc. 8th IEEE Information Visualization Conference (InfoVis'02)*, pages 110–116. IEEE Computer Society, 2002. doi:10.1109/INFVIS.2002.1173155.

Internally-Convex Drawings of Outerplanar Graphs in Small Area

Michael A. Bekos
University of Ioannina, Greece

Giordano Da Lozzo
Roma Tre University, Italy

Fabrizio Frati
Roma Tre University, Italy

Giuseppe Liotta
University of Perugia, Italy

Antonios Symvonis
National Technical University of Athens, Greece

Abstract

A well-known result by Kant [*Algorithmica*, 1996] implies that n-vertex outerplane graphs admit embedding-preserving planar straight-line grid drawings where the internal faces are convex polygons in $O(n^2)$ area. In this paper, we present an algorithm to compute such drawings in $O(n^{1.5})$ area. We also consider outerplanar drawings in which the internal faces are required to be strictly-convex polygons. In this setting, we consider outerplanar graphs whose weak dual is a path and give a drawing algorithm that achieves $\Theta(nk^2)$ area, where k is the maximum size of an internal facial cycle.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Mathematics of computing → Graph theory

Keywords and phrases Grid drawings, convexity, area bounds, outerplanar graphs

Digital Object Identifier 10.4230/LIPIcs.GD.2025.18

Related Version *Full Version*: `https://arxiv.org/abs/2508.19913` [3]

Funding *Bekos* and *Symvonis* were supported by HFRI Grant No: 26320. *Da Lozzo* and *Frati* were supported by the European Union, Next Generation EU, Mission 4, Component 1, CUP J53D23007130006 PRIN proj. 2022ME9Z78 "NextGRAAL: Next-generation algorithms for constrained GRAph visuALization". *Liotta* was supported by MUR PON Proj. ARS01 00540 and by MUR PRIN Project no. 2022TS4Y3N.

1 Introduction

Drawing outerplanar graphs in small area is a core topic in Graph Drawing which has been the subject of intense work. The first one we are aware of dates back to 1980, when Leiserson provided an algorithm for the construction of linear-area VLSI layouts of bounded-degree outerplanar graphs [19]. In the early 90's, de Fraysseix, Pach, and Pollack [10] and Schnyder [20] proved that n-vertex planar graphs, and thus n-vertex outerplanar graphs, admit $O(n^2)$-area planar straight-line grid drawings. This bound is the best possible for planar graphs, however whether a subquadratic area upper bound can be achieved for outerplanar graphs has been an intriguing question asked repeatedly (see, e.g., [5, 7, 14, 17]). A positive answer was first provided by Di Battista and Frati [12]. They presented an algorithm which, given an n-vertex maximal outerplanar graph G with weak dual tree T, uses a structural decomposition of binary trees by Chan [8] in order to construct an $O(n^{1.48})$ area "star-shaped" drawing of T; this is a planar straight-line grid drawing which can be


© Michael A. Bekos, Giordano Da Lozzo, Fabrizio Frati, Giuseppe Liotta, and Antonios Symvonis;
licensed under Creative Commons License CC-BY 4.0



augmented to an outerplanar straight-line drawing of G within the same area bound. The same strategy has later led to an almost-linear area bound, namely $O\left(n2^{\sqrt{2\log n}}\sqrt{\log n}\right)$ [16], and also to an $O(dn \log n)$ bound for outerplanar graphs of maximum degree d [15].

In this paper we study the area requirements of *convex drawings* of outerplanar graphs, in which the boundary of every face (including the outer face) is a convex polygon. It was surprising to us that this question has not been studied before, considering the prominent place that convex drawings occupy in the graph drawing literature. We only mention here the result on convex drawings that is most relevant to our paper. Namely, Chrobak and Kant [9, 18] and, independently, Felsner [13] have presented extensions of the algorithms by de Fraysseix, Pach, and Pollack [10] and Schnyder [20], proving that every 3-connected planar graph admits a convex drawing in quadratic area, which is asymptotically optimal. Specifically, Felsner's upper bound is $(f-1) \times (f-1)$, where f denotes the number of faces.

Strictly-convex drawings, in which no three vertices on the boundary of a face are allowed to be collinear, require larger area. Indeed, an n-vertex cycle cannot be drawn as a strictly-convex polygon on a grid of size $o(n^3)$ [1]. This bound is not yet matched by a corresponding upper bound. Bárány and Rote [2] proved that n-vertex 3-connected planar graphs admit strictly-convex grid drawings in $O(n^4)$ area. Bekos, Gronemann, Montecchiani, and Symvonis [4] provided an alternative algorithm with the same asymptotic area bound but with significantly smaller hidden constants.

The construction of small-area convex grid drawings of outerplanar graphs encounters two natural barriers. First, a planar graph admits a convex drawing only if it is biconnected, hence one is required to consider biconnected outerplanar graphs only. Second, we observe that there exist n-vertex biconnected outerplanar graphs that require $\Omega(n^3)$ area in any convex grid drawing (Section 2); note that an $O(n^3)$ upper bound for (even strictly-)convex drawings of n-vertex biconnected outerplanar graphs can be achieved by drawing the cycle delimiting the outer face as a strictly-convex polygon [1] and drawing the internal edges as chords. We thus relax the convexity requirement and require only the internal faces to be convex, that is, we consider *internally-convex drawings*. For outerplanar graphs, this is a natural variation of the convex drawing style, taking into account that the outer face of an outerplanar graph is *special*: All the vertices are incident to it and this property defines the graph class. Chrobak and Kant's algorithms [9, 18] allow us to obtain internally-convex drawings of n-vertex outerplanar graphs on a grid of size $O(n^2)$ (via a suitable augmentation of the input graph to a 3-connected planar graph, by adding vertices and edges in its outer face). This restores in the internally-convex setting the pursuit for a sub-quadratic area upper bound for grid drawings of outerplanar graphs. We also consider the, in our opinion equally interesting, question of determining the area requirements of *internally-strictly-convex drawings* of outerplanar graphs.

Our contributions. Our main result is an algorithm that constructs an internally-convex grid drawing with $O(n^{1.5})$ area for n-vertex outerplanar graphs (Section 3). We employ some ideas and tools from a recent paper [6]. The main geometric component of our algorithm consists of constructing a "leveled" drawing, where consecutive levels are drawn on consecutive horizontal grid lines, up to a level where there are sufficiently few vertices so that the drawing can "make a turn" and extend horizontally rather than vertically. We remark that ours is the first algorithm that constructs outerplanar straight-line grid drawings in sub-quadratic area by drawing directly the outerplanar graph, without passing through a star-shaped drawing of its weak dual tree. This aspect might be of independent interest.

For internally-strictly-convex grid drawings, the same arguments as in the strictly-convex setting prove that $\Theta(n^3)$ is a tight bound for the area requirements. However, the natural question to consider here is how the area relates not only to the number n of vertices but also to the maximum size k of an internal face. We show that there exist outerplanar graphs requiring $\Omega(nk^2)$ area in any internally-strictly-convex grid drawing and that the lower bound can be matched by a corresponding upper bound for outerpaths, that is, biconnected outerplanar graphs whose weak dual is a path (Section 4).

We conclude in Section 5 with several open problems. Full proofs of statements marked with a ($\star$) can be found in the extended version of the paper [3].

2 Preliminaries

We assume familiarity with Graph Drawing, see, e.g. [11].

A *drawing* of a graph maps each vertex to a distinct point in the plane and each edge to a Jordan arc between its endpoints. In a *grid drawing* vertices have integer coordinates. Let Γ be a grid drawing of a graph. Let $\max_q(\Gamma)$ and $\min_q(\Gamma)$ be the maximum and minimum q-coordinate of a vertex in Γ, with $q \in \{x, y\}$, respectively. The *height* $h(\Gamma)$ and *width* $w(\Gamma)$ of Γ are the positive integers $h(\Gamma) = \max_y(\Gamma) - \min_y(\Gamma) + 1$ and $w(\Gamma) = \max_x(\Gamma) - \min_x(\Gamma) + 1$, respectively. In other words, $h(\Gamma)$ and $w(\Gamma)$ represent the number of horizontal and vertical grid lines intersecting the smallest axis-parallel rectangle enclosing Γ, respectively. The *area* of a drawing is the product of its width and height.

A drawing is *planar* if no two edges intersect, except at a common end-point. In a planar drawing, a vertex or an edge is *external* if it is incident to the outer face, and *internal* otherwise. A graph is *planar* if it admits a planar drawing. A graph is *outerplanar* if it admits a planar drawing in which each vertex is external. A planar drawing partitions the plane into topologically connected regions called *faces*. The unbounded face is the *outer face*, whereas the bounded faces are the *internal faces*. Two planar drawings of a connected graph are *equivalent* if they induce (i) the same counter-clockwise order of the edges incident to each vertex and (ii) the same counter-clockwise order of the edges along the boundaries of their outer faces. The equivalence classes of equivalent outerplanar drawings are called *outerplanar embeddings*, or in this paper just *embeddings*. An *outerplane graph* is a planar graph equipped with an outerplanar embedding. A biconnected outerplanar graph has a unique outerplanar embedding (up to an inversion of all the orders defining the embedding), hence we often talk about faces of a biconnected outerplane graph, referring to faces of its unique outerplanar embedding. Also, a drawing of a graph is *embedding-preserving* if it respects an embedding associated to the graph.

In a *straight-line drawing*, edges are drawn as straight-line segments. In a planar straight-line drawing, a face is *convex* (*strictly-convex*) if it is delimited by a polygon whose internal angles are not larger than 180° (resp. are smaller than 180°). A *convex drawing* (a *strictly-convex drawing*) is a straight-line planar drawing in which every face, including the outer face, is convex (resp., strictly-convex). An *internally-convex drawing* (an *internally-strictly-convex drawing*) is a straight-line planar drawing in which every internal face, but not necessarily the outer face, is convex (resp., strictly-convex). When dealing with outerplanar graphs, we do not require convex drawings to be outerplanar per se. Our results however hold in the strongest possible sense: The lower bounds on the area hold true even for drawings that are not outerplanar, while the upper bounds are proved by constructing outerplanar drawings.

The following theorem motivates the study of internally-convex drawings.

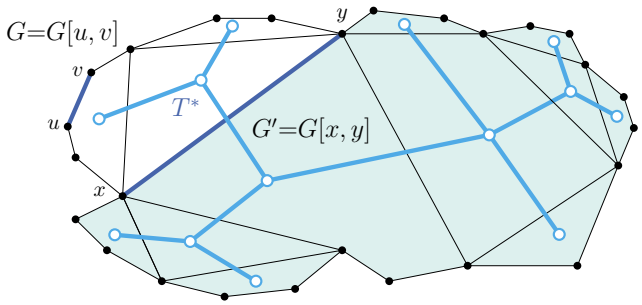


Figure 1 An outerplanar graph $G[u, v]$ rooted at the external edge (u, v) and its weak dual T^*. The outerplanar graph $G' = G[x, y]$ is light-blue shaded.

▶ **Theorem 1.** *There exists an n-vertex biconnected outerplanar graph that requires $\Omega(n^3)$ area in every convex grid drawing.*

Proof. Let G be the n-vertex biconnected outerplanar graph obtained from a cycle $\mathcal{C}$ with $n/2$ vertices by attaching a degree-2 vertex incident to the end-vertices of each edge of $\mathcal{C}$. First, every convex drawing of G respects the unique outerplanar embedding of G, as moving a degree-2 vertex inside $\mathcal{C}$ would create an angle larger than 180° in an internal face at that vertex. Second, in every outerplanar convex drawing of G, the drawing of $\mathcal{C}$ is strictly-convex, as the angle at each vertex v of $\mathcal{C}$ in the interior of the cycle delimiting the outer face of G is at most 180° and part of this angle is used by the two triangular faces incident to v. Since every strictly-convex drawing of a cycle requires cubic area [1], the lower bound follows. ◀

Let G be a biconnected outerplane graph with $n \geq 3$ vertices; refer to Fig. 1. Note that G has at least one internal face. The *weak dual* of G, denoted by T^*, is the tree having a vertex for each internal face of G and an edge between any two vertices corresponding to internal faces sharing an edge. For simplicity, we use the same label to denote a vertex of T^* and the corresponding face of G. If T^* is a path, then G is an *outerpath*. Let (u, v) be an external edge of G such that v immediately precedes u in counter-clockwise order along the outer face of G. The outerplane graph G with the designated edge (u, v), denoted by $G[u, v]$, is said to be *rooted at* (u, v). Let (x, y) be an internal edge of $G[u, v]$ such that u, x, y, v appear in this counter-clockwise order along the outer face of G (possibly $u = x$ or $y = v$ may hold). Consider the outerplane subgraph G' of G induced by the vertices x, y, and the vertices that are encountered when traversing the outer face of G in counter-clockwise direction from x to y. Observe that the edge (x, y) is an external edge of G' such that y immediately precedes x in counter-clockwise order along the outer face of G'. Then we denote by $G[x, y]$ the outerplanar graph G' rooted at the edge (x, y).

Let G be rooted at (u, v) and let f_1 be the internal face of G incident to (u, v). The *extended weak dual tree* T of $G[u, v]$ is the tree with root f_1 obtained from the weak-dual tree T^* of G by inserting a new leaf ℓ_e for each external edge e of G different from (u, v) and an edge (f, ℓ_e) if f is the internal face of G incident to e. Let d be the number of vertices of T. Note that $d \geq n$, as T contains the root f_1 plus one leaf for each of the $n - 1$ external edges of G different from (u, v). Also, $d \leq 2n - 3$, since G has $n - 1$ external edges different from (u, v) and at most $n - 2$ internal faces. Let $\pi = (f_1, f_2, \ldots, f_p)$ be a root-to-leaf path in T. We denote by $G[\pi]$ the outerpath induced by the vertices incident to the faces $f_1, f_2, \ldots, f_{p-1}$. We call $G[\pi]$ the *outerpath dual to* π. Denote by e_p the edge of G dual to (f_{p-1}, f_p). Let $u = z_1, z_2, \ldots, z_a, z_{a+1}, \ldots, z_m = v$ be the counter-clockwise order of the

vertices along the outer face of $G[\pi]$, where $e_p = (z_a, z_{a+1})$. We call the rooted outerplanar graphs $G[z_1, z_2], \ldots, G[z_{a-1}, z_a]$ *left subgraphs of* G *at* π and the rooted outerplanar graphs $G[z_{a+1}, z_{a+2}], \ldots, G[z_{m-1}, z_m]$ *right subgraphs of* G *at* π.

3 Internally-Convex Drawings

In this section we prove the following theorem, which is our main result.

▶ **Theorem 2.** *Every n-vertex outerplane graph admits an embedding-preserving internally-convex grid drawing in $O(n^{1.5})$ area.*

Our proof is inspired by the proof that every outer-1-plane graph admits an embedding-preserving visibility representation in $O(n^{1.5})$ area [6]. In particular, we are going to use the following theorem, which generalizes a well-known result by Chan [8].

▶ **Theorem 3** ([6, Theorem 1]). *Let $p = 0.48$. Given any rooted tree with n vertices, there exists a root-to-leaf path π such that for any left subtree α and for any right subtree β of π, $|\alpha|^p + |\beta|^p \leq (1-\delta)n^p$, for some constant $0 < \delta < 1$.*

The following lemma is similar to the combination of Definition 1 and Observation 3 from [6], although with different constants. Recall that $p = 0.48$ and $0 < \delta < 1$ are the constants from Theorem 3. We define the constant c as $c := 2/\delta$.

▶ **Lemma 4** ($\star$). *Let $f(d)$ be the recursive function defined on $\mathbb{N}_0$ as follows. First, $f(0) = 0$ and $f(1) = 1$. Second, for $d \in \mathbb{N}_0$ with $d \geq 2$, we have that*

$$f(d) = \max_{d_a, d_b \in \mathbb{N}_0 : d_a^p + d_b^p \leq (1-\delta)d^p} \{f(d_a) + f(d_b)\} + 2\sqrt{d}.$$

Then $f(d) \leq c\sqrt{d}$.

Let G be an n-vertex outerplane graph; refer to Fig. 2. In what follows, we assume G to be biconnected. Indeed, if it is not, it can be augmented to a biconnected outerplane graph G' by introducing edges in its outer face. Then the restriction of an embedding-preserving internally-convex drawing Γ' of G' yields an embedding-preserving internally-convex drawing of G that inherits the area bounds of Γ'. Let u and v be any two vertices such that v immediately precedes u in counter-clockwise order along the outer face of G. We root G at the edge (u, v). Recall that $G[u, v]$ denotes the outerplanar graph G rooted at the edge (u, v).

We show that $G[u, v]$ admits a (small-area) *uv-separated drawing*. This is an embedding-preserving internally-convex grid drawing Γ of G satisfying the following properties:

- **P.1** u and v lie on a horizontal line h_0;
- **P.2** all neighbors of u and v lie on the horizontal line h_1 one unit below h_0;
- **P.3** all the other vertices of G lie not above h_1; and
- **P.4** all vertices different from u (from v) are to the right of u (resp. to the left of v).

The following property is a direct consequence of the definition of uv-separated drawing.

▶ **Property 1.** *Let Γ be a uv-separated drawing of $G[u, v]$. Vertices u and v can be shifted arbitrarily by integer amounts to the left and right, respectively, while maintaining Γ an outerplanar internally-convex grid drawing.*

If $n \geq 3$, then let T be the extended weak dual tree of $G[u, v]$ and let d be the number of vertices of T. If $n = 2$, then T is not defined and we set $d = 1$. This choice is justified by the size of the subtrees of T we generate. Indeed, if $n \geq 3$, we will select a root-to-leaf path π

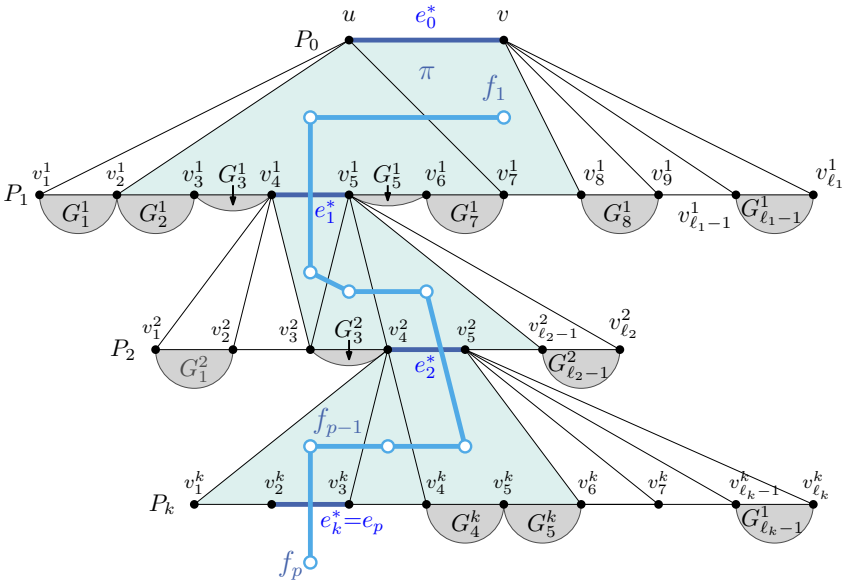


Figure 2 An outerplanar graph $G[u, v]$ rooted at the external edge (u, v). The outerpath $G[\pi]$ dual to the path π selected by Theorem 3 is light-blue shaded. Transition edges are drawn thick blue.

in T and consider left and right subgraphs of G at π. If any of such subgraphs is just an edge, it is associated with a subtree of T which consists of a single vertex, from which the correspondence between $n = 2$ and $d = 1$ stems.

We show an algorithm that, by induction on n, constructs a uv-separated drawing Γ of $G[u, v]$ satisfying the following two properties:

- *Property W:* every vertical grid line intersecting Γ contains at least one vertex of G (and thus the width of Γ is at most n); and
- *Property H:* every horizontal grid line intersecting Γ contains at least one vertex of G and the height of Γ is at most $f(d)$.

The statement implies Theorem 2, given that $f(d) \in O(\sqrt{d})$ by Lemma 4 and since $d \leq 2n-3$.

Denote by $h(d)$ the maximum value of $h(\Gamma)$, over all drawings Γ constructed by our algorithm for rooted outerplane graphs whose extended weak dual has d vertices.

In the base case, we have $n = 2$ (and thus $d = 1$). Hence, G is the edge (u, v) and a drawing Γ satisfying *Properties W* and *H* is constructed by placing u on a grid point one unit to the left of v. In particular, note that $h(\Gamma) = 1$ and that $f(1) = 1$.

In the inductive case, we have $n > 2$, hence G has internal faces, since it is biconnected. Let f_1 be the root of T, which is the internal face of G incident to (u, v). Let $\pi = (f_1, f_2, \ldots, f_p)$ be the root-to-leaf path in T from Theorem 3, where e_p is the edge of G dual to (f_{p-1}, f_p).

We introduce some notation and definitions. Call $e_0^* = (u, v)$ a *transition edge*; let $V_0 := \{u, v\}$, let the path $P_0 = (v_0^0, v_1^0)$ coincide with (u, v), and let the rooted outerplane graph G_0^* coincide with $G[u, v]$. Suppose that, for some $i \geq 1$, a transition edge e_{i-1}^* has been defined that is part of a path P_{i-1}. Consider the set of vertices that do not belong to $V_0 \cup \cdots \cup V_{i-1}$ and that belong to the faces of G incident to the end-vertices of e_{i-1}^*; these vertices induce a path $P_i = (v_1^i, \ldots, v_{\ell_i}^i)$, where the labels of the vertices are such that $u, v_1^i, v_{\ell_i}^i, v$ appear in this counter-clockwise order along the outer face of G. See again Fig. 2, where, for example, P_2 is the path induced by the vertices that do not belong to $V_0 \cup V_1$ (roughly speaking, by the vertices that are on the "correct side" of e_1^*) and that

belong to the faces of G incident to the end-vertices of e_1^* (only one of such faces is incident to both the end-vertices of e_1^*, the other ones are incident to a single end-vertex of e_1^*). Only one of the edges of P_i might be dual to an edge of π; this edge of P_i is the transition edge e_i^* with end-vertices $v_{s_i}^i$ and $v_{s_i+1}^i$, for some $1 \leq s_i \leq \ell_i - 1$. The set V_i comprises the vertices $v_1^i, \ldots, v_{\ell_i}^i$ of P_i, as well as the vertices of the subgraphs $G_j^i := G[v_j^i, v_{j+1}^i]$ of G, for $j = 1, \ldots, \ell_i - 1$ with $j \neq s_i$. Observe that, for $j = 1, \ldots, s_i - 1$, the graph G_j^i is a left subgraph of G at π or a proper subgraph of a left subgraph of G at π. Similarly, for $j = s_i + 1, \ldots, \ell_i - 1$, the graph G_j^i is a right subgraph of G at π or a proper subgraph of a right subgraph of G at π. We denote by G_i^* the graph $G[v_{s_i}^i, v_{s_i+1}^i]$. For the last-defined path P_k, we have that either the transition edge e_k^* does not exist or, if it does, it is the edge e_p, which is incident to the outer face of G and thus G_k^* coincides with the edge e_k^*. An example of a situation in which e_k^* does not exist is the one in which P_k is a single vertex. This happens if f_{p-1}, the second-to-last vertex of π, is a triangular face delimited by the end-vertices of e_{k-1}^* and by one additional vertex, which coincides with P_k. In this case e_p, the edge of G dual to the last edge (f_{p-1}, f_p) of π, connects a vertex of e_{k-1}^* with the unique vertex of P_k and e_k^* does not exist.

Ideally, we would like the uv-separated drawing we construct to have the following features. The vertices $v_1^i, \ldots, v_{\ell_i}^i$ of each path P_i should appear on a common horizontal line h_i in this left-to-right order, and h_i should lie one unit above h_{i+1}. The subgraphs G_j^i with $j \neq s_i$ should be recursively drawn and placed below h_i, except for the edge (v_j^i, v_{j+1}^i) which lies on h_i, in such a way that any two of these subgraphs are *horizontally disjoint*, that is, there exists a vertical line that keeps the drawings of the two subgraphs on different sides, except, possibly, for a single vertex shared by the two subgraphs which lies on the line. The subgraph $G_{s_i}^i$ is not drawn recursively, as it coincides with the subgraph G_i^* on which the level-by-level construction we are describing is further applied. Recall that k is the index of the path P_k defined last in the procedure discussed above. If $k \leq \sqrt{n}$ (**Case 1**), this plan can be accomplished. Otherwise (**Case 2**), by a simple vertex counting argument, there exists an index $2 \leq t \leq 1 + \sqrt{n}$ such that the set V_t has at most $\sqrt{n}$ vertices. In this case, we draw the vertices in V_t differently than before, so that the transition edge e_t^* is vertical. The small size of V_t allows us to "turn" from horizontal to vertical without increasing the height dramatically. From e_t^* towards the end of π, the construction proceeds by drawing the vertices of the outerpath dual to π on two horizontal lines, and by placing the recursively-constructed drawings of the right and left subgraphs of G at π above and below it, respectively.

We now formalize this argument. As sketched above, we distinguish two cases. In **Case 1**, we have $k \leq \sqrt{n}$; refer to Fig. 3. For $i = 1, \ldots, k$ and for $j = 1, \ldots, \ell_i - 1$ with $j \neq s_i$, we recursively construct a $v_j^i v_{j+1}^i$-separated drawing Γ_j^i of $G_j^i[v_j^i, v_{j+1}^i]$. We construct, for $i = k-1, k-2, \ldots, 0$, a $v_{s_i}^i v_{s_i+1}^i$-separated drawing Γ_i^* of $G_i^*[v_{s_i}^i, v_{s_i+1}^i]$ as described in the following. Notice that the order of the indices ensures that, if G_{i+1}^* exists, then its drawing Γ_{i+1}^* has been already constructed once Γ_i^* is about to be constructed. In order to start the iteration, if e_k^* does not exist, then G_k^* does not exist either, and hence Γ_k^* does not need to be defined. If e_k^* exists, then G_k^* coincides with such an edge and Γ_k^* is constructed by placing $v_{s_k}^k$ on a grid point one unit to the left of $v_{s_k+1}^k$. In order to construct Γ_i^*, we place the recursively constructed drawings Γ_j^{i+1} with $j = 1, \ldots, \ell_{i+1} - 1$ with $j \neq s_{i+1}$, as well as the drawing Γ_{i+1}^* (if G_{i+1}^* exists), side by side, so that the placement of a vertex v_j^{i+1} coincides in two drawings it belongs to; these are Γ_{j-1}^{i+1} and Γ_j^{i+1} if v_j^{i+1} is neither $v_{s_{i+1}}^{i+1}$ nor $v_{s_{i+1}+1}^{i+1}$, or are $\Gamma_{s_{i+1}-1}^{i+1}$ and Γ_{i+1}^* if v_j^{i+1} is $v_{s_{i+1}}^{i+1}$, or are Γ_{i+1}^* and $\Gamma_{s_{i+1}+1}^{i+1}$ if v_j^{i+1} is $v_{s_{i+1}+1}^{i+1}$. This ensures that the vertices of P_{i+1} all lie on the same horizontal line h_{i+1}. Further, we

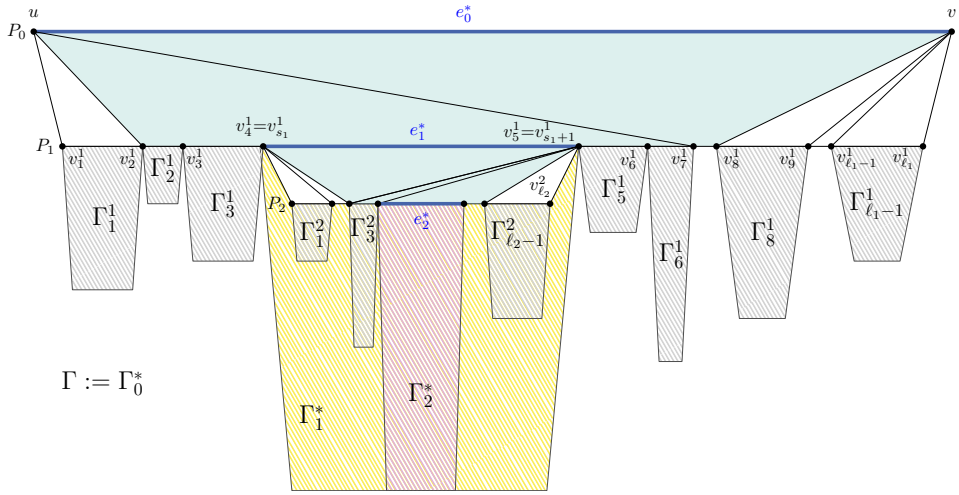


Figure 3 Construction for a uv-separated drawing of $G[u, v]$ when $k \leq \sqrt{n}$. To avoid cluttering, vertical and horizontal proportions are not respected, but vertices are assumed to lie on the grid.

place $v^i_{s_i}$ one unit above and one unit to the left of the leftmost vertex of P_{i+1}, and $v^i_{s_i+1}$ one unit above and one unit to the right of the rightmost vertex of P_{i+1}. This results in the desired $v^i_{s_i}v^i_{s_i+1}$-separated drawing Γ^*_i of $G[v^i_{s_i}, v^i_{s_i+1}]$. When $i = 0$, the resulting drawing $\Gamma := \Gamma^*_0$ is a uv-separated drawing of $G[u, v]$, as established by the following lemma.

▶ **Lemma 5** (⋆). *In Case 1, the constructed drawing* Γ *is a* uv*-separated drawing of* $G[u, v]$ *satisfying Properties W and H.*

We now describe the construction of **Case 2**. Recall that, in this case, we have $k \geq 1+\sqrt{n}$ We start the description with the following simple combinatorial lemma.

▶ **Lemma 6.** *There exists an index* t *with* $2 \leq t \leq 1 + \sqrt{n}$ *such that* $|V_t| \leq \sqrt{n}$.

Proof. Since the sets $V_2, V_3, \ldots, V_{1+\sqrt{n}}$ are disjoint and since there are $\sqrt{n}$ of them, the smallest among them contains at most $\sqrt{n}$ vertices. ◀

The construction for Case 2 proceeds in three steps. In **Step 1**, we construct a drawing Γ^*_t of G^*_t; in **Step 2**, we augment Γ^*_t to a drawing Γ^*_{t-1} of G^*_{t-1}; finally, in **Step 3**, we augment Γ^*_{t-1} to the desired drawing Γ of G.

Step 1. We construct a drawing Γ^*_t of G^*_t; refer to Fig. 4. Recall that the transition edge e^*_t has end-vertices $v^t_{s_t}$ and $v^t_{s_t+1}$ and that G^*_t is the graph $G[v^t_{s_t}, v^t_{s_t+1}]$. Let (f_j, f_{j+1}) be the edge of π dual to e^*_t, let π' be the subpath $(f_{j+1}, f_{j+2}, \ldots, f_p)$ of π, and let $(u_0 = v^t_{s_t}, u_1, \ldots, u_x, w_y, w_{y-1}, \ldots, w_1, w_0 = v^t_{s_t+1})$ be the counter-clockwise order of the vertices along the outer face of $G[\pi']$, where (u_x, w_y) is the edge e_p. We initialize Γ^*_t by drawing e^*_t as a vertical segment of height 1. For $j = 1, \ldots, x$, we recursively construct a drawing Λ^u_j of $G[u_{j-1}, u_j]$. We place the drawings $\Lambda^u_1, \ldots, \Lambda^u_x$ side by side, so that the placement of a vertex u_j coincides in the drawings Λ^u_j and Λ^u_{j+1} where it appears, for $j = 1, \ldots, x-1$, and so that the placement of u_0 coincides in Λ^u_1 and in the drawing of e^*_t. Analogously, for $j = 1, \ldots, y$, we recursively construct a drawing Λ^w_j of $G[w_j, w_{j-1}]$. Unlike for $\Lambda^u_1, \ldots, \Lambda^u_x$, we rotate the drawings $\Lambda^w_1, \ldots, \Lambda^w_y$ by 180°, and place them side by side, so that the placement of a vertex w_j coincides in Λ^w_j and Λ^w_{j+1}, for $j = 1, \ldots, y-1$, and so that the placement of w_0 coincides in Λ^w_1 and in the drawing of e^*_t. This completes the construction of Γ^*_t.

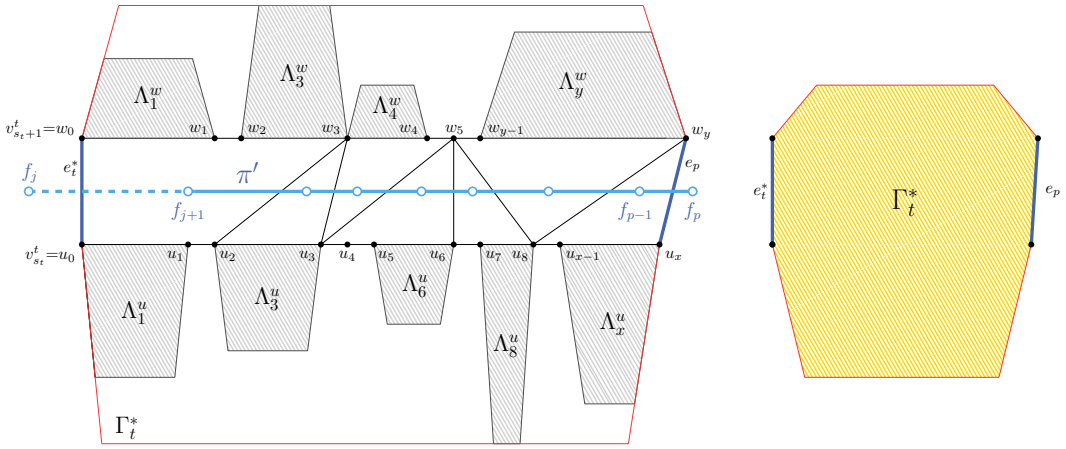


Figure 4 (Left) Construction of the drawing Γ_t^* of G_t^*. Labels Λ_j^u and Λ_j^v are omitted for single edges. (Right) Polygon used to represent the drawing of Γ_t^* in the subsequent figures.

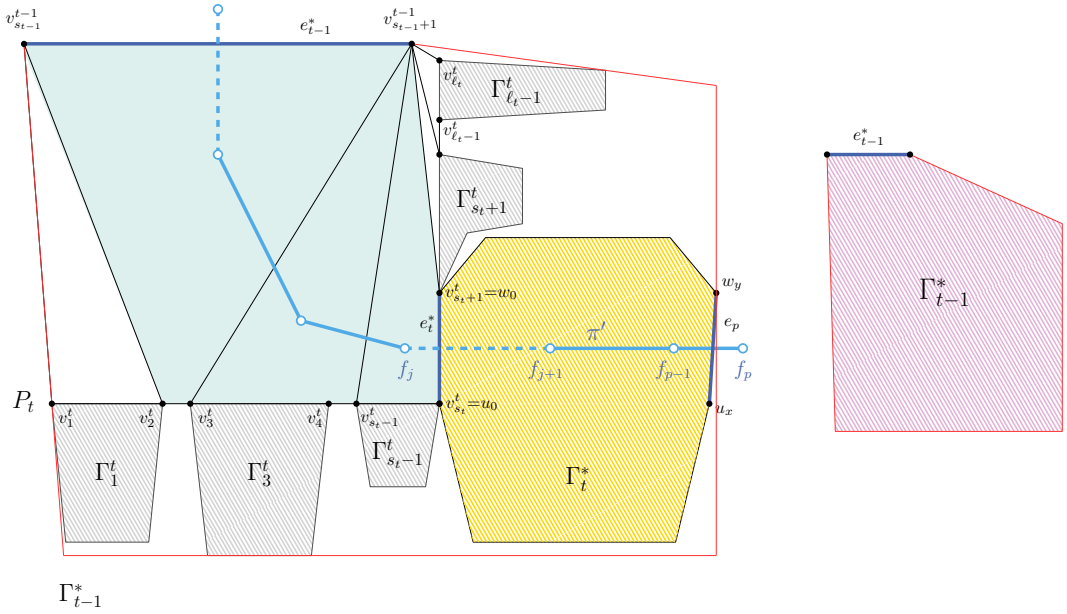


Figure 5 (Left) Construction of Γ_{t-1}^* from Γ_t^*, whose boundary is depicted by the yellow-tiled region. (Right) Polygon used to represent Γ_{t-1}^* in the subsequent figures.

Step 2. We now augment Γ_t^* to a drawing Γ_{t-1}^* of G_{t-1}^*; refer to Fig. 5. Recall that the path P_t has vertices $(v_1^t, v_2^t, \ldots, v_{s_t}^t, v_{s_t+1}^t, \ldots, v_{\ell_t}^t)$. For $j = 1, \ldots, \ell_t - 1$ with $j \neq s_t$, we recursively construct a $v_j^t v_{j+1}^t$-separated drawing Γ_j^t of $G[v_j^t, v_{j+1}^t]$. We place the recursively constructed drawings $\Gamma_1^t, \ldots, \Gamma_{s_t-1}^t$ side by side, so that the placement of a vertex v_j^t coincides in Γ_{j-1}^t and Γ_j^t, for $j = 2, \ldots, s_t - 1$, and so that the placement of $v_{s_t}^t$ coincides in $\Gamma_{s_t-1}^t$ and in Γ_t^*. Next, we rotate the drawings $\Gamma_{s_t+1}^t, \ldots, \Gamma_{\ell_t-1}^t$ in counter-clockwise direction by 90°, and we stack them one on top of the other, so that the placement of a vertex v_j^t coincides in Γ_{j-1}^t and Γ_j^t, for $j = s_t + 2, \ldots, \ell_t - 1$, and so that the placement of $v_{s_t+1}^t$ in $\Gamma_{s_t+1}^t$ is on the same vertical line as in Γ_t^* and on the same horizontal line as the highest vertex in Γ_t^* (refer to the left side of Fig. 6). This might provide a double placement for $v_{s_t+1}^t$ (unless $v_{s_t+1}^t$ is the highest vertex in Γ_t^*). The issue is resolved by keeping for $v_{s_t+1}^t$ the position it

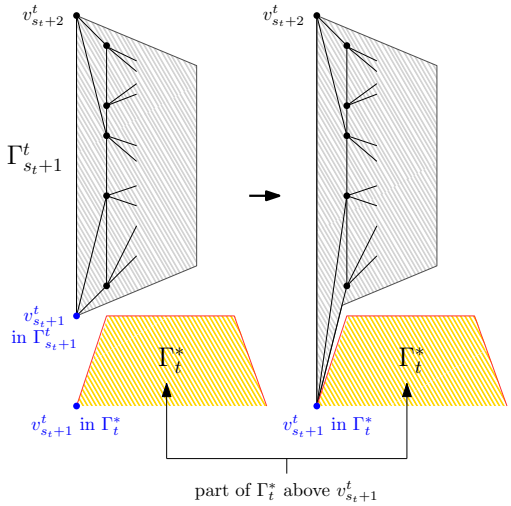


Figure 6 Transformation of $\Gamma^t_{s_t+1}$ shifting downward $v^t_{s_t+1}$ while avoiding overlapping with Γ^*_t.

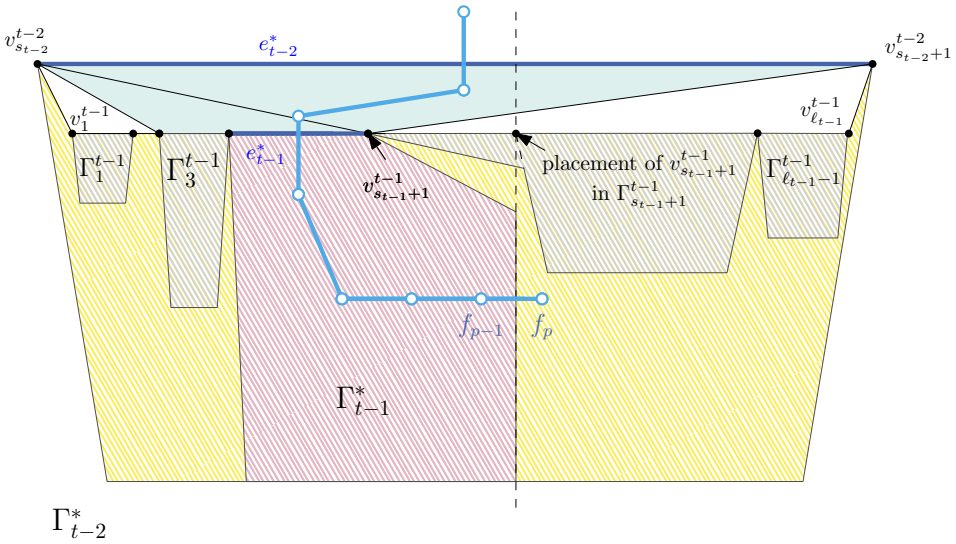


Figure 7 Construction of Γ^*_{t-2} from Γ^*_{t-1}, whose boundary is depicted by the red-tiled region. The illustration shows the modification of the placement of $v^{t-1}_{s_{t-1}+1}$ from the one it has in $\Gamma^{t-1}_{s_{t-1}+1}$.

has in Γ^*_t; note that, in $\Gamma^t_{s_t+1}$, this amounts to shifting $v^t_{s_t+1}$ downwards (refer to the right side of Fig. 6). The drawing of Γ^*_{t-1} is completed by placing $v^{t-1}_{s_{t-1}}$ and $v^{t-1}_{s_{t-1}+1}$ on the same horizontal line, one unit higher than the highest vertex in Γ^*_{t-1} so far (this is either $v^t_{\ell_t}$ if $s_t \leq \ell_t - 2$ or the highest vertex in Γ^*_t otherwise), so that $v^{t-1}_{s_{t-1}+1}$ is one unit to the left of the vertical line through e^*_t and $v^{t-1}_{s_{t-1}}$ is one unit to the left of the leftmost vertex in Γ^*_{t-1} so far (this is either v^t_1 if $s_t \geq 2$ or $v^{t-1}_{s_{t-1}+1}$ otherwise).

Step 3. Finally, we show how to augment Γ^*_{t-1} to a uv-separated drawing of $G[u, v]$. This is done similarly to Case 1. Namely, we construct, for $i = t-2, t-3, \ldots, 0$, a $v^i_{s_i} v^i_{s_i+1}$-separated drawing Γ^*_i of $G[v^i_{s_i}, v^i_{s_i+1}]$ from an already constructed $v^{i+1}_{s_{i+1}} v^{i+1}_{s_{i+1}+1}$-separated drawing Γ^*_{i+1} of $G[v^{i+1}_{s_{i+1}}, v^{i+1}_{s_{i+1}+1}]$. Recall that, in Step 2, we constructed Γ^*_{t-1}. As in Case 1, we place the

recursively constructed drawings Γ_j^{i+1} with $j = 1, \ldots, \ell_{i+1} - 1$ with $j \neq s_{i+1}$, as well as the drawing Γ_{i+1}^*, side by side, so that the placement of a vertex v_j^{i+1} coincides in two drawings it belongs to, with one exception in the case $i = t - 2$; refer to Fig. 7. Namely, the drawing $\Gamma_{s_{t-1}+1}^{t-1}$ of $G_{s_{t-1}+1}^{t-1}$ is embedded so that $v_{s_{t-1}+1}^{t-1}$ is on the same horizontal line as in Γ_{t-1}^* and on the same vertical line as the rightmost vertex in Γ_{t-1}^*. This provides a double placement for $v_{s_{t-1}+1}^{t-1}$. The issue is resolved by keeping for $v_{s_{t-1}+1}^{t-1}$ the position it has in Γ_{t-1}^*; note that, in $\Gamma_{s_{t-1}+1}^{t-1}$, this amounts to shifting $v_{s_{t-1}+1}^{t-1}$ leftwards. The desired $v_{s_i}^i v_{s_i+1}^i$-separated drawing Γ_i^* of $G[v_{s_i}^i, v_{s_i+1}^i]$ is completed by placing $v_{s_i}^i$ one unit above and one unit to the left of the leftmost vertex of P_{i+1}, and $v_{s_i+1}^i$ one unit above and one unit to the right of the rightmost vertex of P_{i+1}. When $i = 0$, the obtained drawing $\Gamma := \Gamma_0^*$ is a uv-separated drawing of $G[u, v]$, as proved in the following.

▶ **Lemma 7** (⋆). *In Case 2, the constructed drawing* Γ *is a* uv*-separated drawing of* $G[u, v]$ *satisfying* Property W *and* H.

Sketch. First, it trivially comes from the construction that Γ is a straight-line grid drawing.

Most of the arguments used to prove that Γ is planar and satisfies Properties P.1–P.4 of a uv-separated drawing are easy geometric considerations. For example, each graph $G[u_i, u_{i+1}]$ does not cross any graph $G[u_j, u_{j+1}]$, as it is horizontally disjoint from it, does not cross any subgraph $G[w_j, w_{j+1}]$ as it is vertically disjoint from it, and does not cross $G[\pi']$ as it is vertically disjoint from it, except for the edge (u_i, u_{i+1}) which is shared by the two graphs. The most interesting part in the proof of the planarity of Γ consists of showing that the drawings $\Gamma_{s_t+1}^t$ and Γ_t^* do not cross each other. This proof uses the fact that $\Gamma_{s_t+1}^t$ satisfies the properties of a $v_{s_t+1}^t v_{s_t+2}^t$-separated drawing. In fact, the avoidance of crossings between $\Gamma_{s_t+1}^t$ and Γ_t^* is the driving force behind the definition of our drawing invariant. Namely, by Property P.4 and by the 90° counter-clockwise rotation of $\Gamma_{s_t+1}^t$, we have that $v_{s_t+1}^t$ is the lowest vertex in $\Gamma_{s_t+1}^t$. Since $v_{s_t+1}^t$ is on the same horizontal line as the highest vertex in Γ_t^*, the only edges of $G_{s_t+1}^t$ that might cross G_t^* in Γ are those incident to $v_{s_t+1}^t$. However, since all the vertices of $G_{s_t+1}^t$ different from $v_{s_t+1}^t$ lie above Γ_t^* and all the neighbors of $v_{s_t+1}^t$ in $G_{s_t+1}^t$ lie on the vertical line one unit to the right of $v_{s_t+1}^t$, by Property P.2 and by the rotation of $\Gamma_{s_t+1}^t$, it follows that the edges of $G_{s_t+1}^t$ incident to $v_{s_t+1}^t$ lie above every edge of G_t^* with which they have a horizontal overlap. Note that, by Property 1, placing $v_{s_t+1}^t$ at the point where it is placed in Γ_t^* maintains the planarity of $\Gamma_{s_t+1}^t$.

The convexity of each internal face f of G in Γ follows by induction if f is internal to some recursively drawn subgraph G_j^i of G. If f is an internal face of $G[\pi']$ or has vertices on two distinct paths among $P_0, P_1, \ldots, P_{t-1}$, then f is drawn as a triangle or a trapezoid in Γ, hence it is convex. The only faces that do not fit in any of these two categories are those incident to $v_{s_{t-1}}^{t-1}$ and/or $v_{s_{t-1}+1}^{t-1}$ and incident to at least one vertex in P_t. The convexity of these faces follows from two facts. First, the polygon Q delimited by the edge $(v_{s_{t-1}}^{t-1}, v_{s_{t-1}+1}^{t-1})$ and by the path P_t is a convex pentagon $(v_{s_{t-1}}^{t-1}, v_{s_{t-1}+1}^{t-1}, v_{\ell_t}^t, v_{s_t}^t, v_1^t)$ (if $s_t > 1$) or a convex quadrilateral $(v_{s_{t-1}}^{t-1}, v_{s_{t-1}+1}^{t-1}, v_{\ell_t}^t, v_1^t)$ (if $s_t = 1$). Second, the faces incident to $v_{s_{t-1}}^{t-1}$ and/or $v_{s_{t-1}+1}^{t-1}$ and incident to at least one vertex in P_t form a convex subdivision of Q.

Property W follows by induction and since every vertical grid line intersecting Γ intersects a recursively constructed drawing or a vertex directly placed by the algorithm.

Concerning *Property H*, we prove that $h(\Gamma)$ is at most t, plus $\sqrt{n}$, plus the maximum height of a recursively constructed drawing of a subgraph of a left subgraph of G at π, plus the maximum height of a recursively constructed drawing of a subgraph of a right subgraph of G at π. The first term accounts for the horizontal grid lines $h_0, h_1, \ldots, h_{t-1}$ on which $P_0, P_1, \ldots, P_{t-1}$ are placed. The $\sqrt{n}$ term accounts for the horizontal grid lines

between the highest line intersecting $\Gamma^t_{\ell_t-1}$ (which is one unit below h_{t-1}) and the lowest line intersecting $\Gamma^t_{s_t+1}$ before $v^t_{s_t+1}$ is moved to the position it has in Γ^*_t. Indeed, since the drawings $\Gamma^t_{s_t+1}, \Gamma^t_{s_t+2}, \ldots, \Gamma^t_{\ell_t-1}$ are constructed recursively, and hence satisfy *Property W*, and are rotated by 90°, and since the graphs $G^t_{s_t+1}, G^t_{s_t+2}, \ldots, G^t_{\ell_t-1}$ have a total of at most $\sqrt{n}$ vertices, by Lemma 6, it follows that the drawings $\Gamma^t_{s_t+1}, \Gamma^t_{s_t+2}, \ldots, \Gamma^t_{\ell_t-1}$ intersect, before moving $v^t_{s_t+1}$, at most $\sqrt{n}$ grid lines. The third and fourth terms in the upper bound for $h(\Gamma)$ account for the total height of the recursively drawn subgraphs different from $G^t_{s_t+1}, G^t_{s_t+2}, \ldots, G^t_{\ell_t-1}$. The key observation here is that, apart for the drawings of $\Gamma^t_{s_t+1}, \Gamma^t_{s_t+2}, \ldots, \Gamma^t_{\ell_t-1}$, which are already accounted for, any two recursively constructed drawings which are stacked one on top of the other are drawings of subgraphs of a left subgraph of G at π and of a right subgraph of G at π (and not of two left subgraphs or of two right subgraphs); in fact, only the right subgraphs $G[w_j, w_{j+1}]$ of G at π are forced to be stacked on top of the left subgraphs $G[u_j, u_{j+1}]$ of G at π (and on top of the graphs $G[v^t_j, v^t_{j+1}]$ with $j < s_t$ which are also subgraphs of left subgraphs of G at π).

Note that $t \leq \sqrt{n}$ by construction and hence $t \leq \sqrt{d}$, given that $d \geq n$. Assume that i and j are indices such that G^i_j is a recursively drawn subgraph of a left subgraph of G at π and $h(\Gamma^i_j)$ is maximum. Indices i' and j' are defined analogously for the right subgraphs of G at π. Let d^i_j and $d^{i'}_{j'}$ be the number of vertices in the extended weak dual trees of G^i_j and $G^{i'}_{j'}$, respectively. By the upper bound proved above, we have $h(\Gamma) \leq 2\sqrt{d} + h(\Gamma^i_j) + h(\Gamma^{i'}_{j'})$ and thus, by induction, $h(\Gamma) \leq 2\sqrt{d} + f(d^i_j) + f(d^{i'}_{j'})$. By Theorem 3, it holds true that $(d^i_j)^p + (d^{i'}_{j'})^p \leq (1-\delta)d^p$. By the definition of $f(d)$ in Lemma 4 (with $d_a = d^i_j$ and $d_b = d^{i'}_{j'}$), we have that $2\sqrt{d} + f(d^i_j) + f(d^{i'}_{j'}) \leq f(d)$, hence $h(\Gamma) \leq f(d)$, which proves *Property H*. ◀

4 Internally-Strictly-Convex Drawings of Outerpaths

In this section, we prove a tight bound on the area required by internally-strictly-convex grid drawings of outerpaths. More precisely, we prove that every n-vertex outerpath whose internal faces have size at most k admits an internally-strictly-convex grid drawing in area $O(nk^2)$, which we also show to match a general lower bound for the internally-strictly-convex grid drawings of outerplanar graphs. We start by proving the lower bound.

▶ **Theorem 8.** *Every n-vertex outerplane graph with $\Omega(\frac{n}{k})$ internal faces of size k requires $\Omega(nk^2)$ area in any internally-strictly-convex grid drawing.*

Proof. Let G be an n-vertex outerplane graph with $\Omega(\frac{n}{k})$ internal faces of size k. First, note that any internally-strictly-convex grid drawing of G must be outerplanar. Indeed, any embedding of an outerplanar graph which is not an outerplanar embedding contains a degree-2 vertex as an internal vertex. However, such a vertex would create an angle larger than or equal to 180° in an internal face. The area lower bound is obtained by a simple packing argument. Since any internally-strictly-convex grid drawing of G is outerplanar, it contains $\Omega(\frac{n}{k})$ internal faces of size k. Each of these faces occupies $\Omega(k^3)$ area [1], hence the total area of the drawing is $\Omega(\frac{n}{k} \cdot k^3)$, which gives the bound claimed by the theorem. ◀

In the rest of the section, we prove a matching upper bound for the case of outerpaths. Note that there exist n-vertex outerpaths with $\Omega(\frac{n}{k})$ internal faces of size k, hence the lower bound of Theorem 8 applies to outerpaths as well. We introduce some notation and definitions. Let G be an outerpath and let $\pi = (f_1, f_2, \ldots, f_{p-1})$ be the weak-dual of G, which is a path; refer to Fig. 8. Let $\hat{e}$ be the edge of G dual to the edge (f_1, f_2) of π and let $\tilde{e}$ be the edge of G dual to the edge (f_{p-2}, f_{p-1}) of π. Moreover, let $\hat{g} = (u_0, v_0)$ be any of

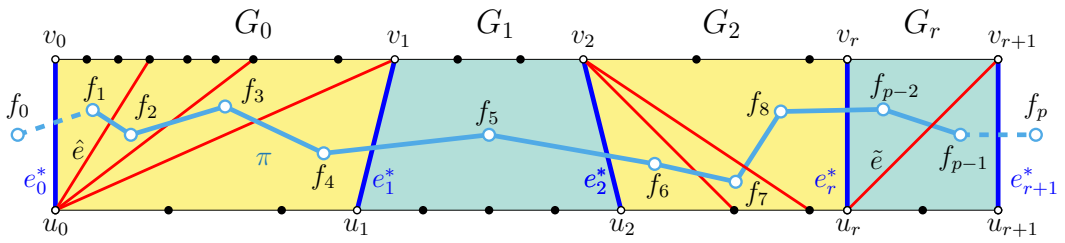


Figure 8 Decomposition of an outerpath G in fan graphs $G_0, G_1, \ldots, G_r$. Gate edges are blue.

the two external edges incident to $\hat{e}$ that belong to the face f_1 of G, where u_0 immediately follows v_0 when traversing the outer face of G in counter-clockwise order. Also, let $\tilde{g}$ be any of the two external edges incident to $\tilde{e}$ that belong to the face f_{p-1} of G. We augment π with two new vertices f_0 and f_p, and edges (f_0, f_1) and (f_{p-1}, f_p). We interpret (f_0, f_1) and (f_{p-1}, f_p) as edges dual to $\hat{g}$ and $\tilde{g}$, respectively.

We now describe a decomposition of G into a sequence of smaller outerpaths such that any two consecutive outerpaths in the sequence share exactly one internal edge of G. We let e_0^* be the edge $\hat{g}$. Let G_0 be the plane graph induced by the vertices delimiting the internal faces of G incident to the end-vertex common to e_0^* and $\hat{e}$ (see the vertex u_0 in Fig. 8). Let e_1^* be the external edge of G_0, other than e_0^*, dual to an edge of π. We call G_0 a *fan graph* with *gate edges* e_0^* and e_1^*. Suppose that, for some $i \geq 1$, a fan graph G_{i-1} with gate edges e_{i-1}^* and e_i^* has been defined. Let u_i and v_i be the end-vertices of e_i^*, where u_i is encountered before v_i when traversing the outer face of G in counter-clockwise direction from u_0 to v_0. We define the fan graph G_i as follows. Let V_{i-1} be defined as $\bigcup_{j=0}^{i-1} V(G_j) \setminus \{u_i, v_i\}$. Then G_i is the outerplane graph induced by the vertices that do not belong to V_{i-1} and that are incident to faces that have u_i or v_i on their boundary. Let e_{i+1}^* be the external edge of G_i, other than e_i^*, dual to an edge of π. The edges e_i^* and e_{i+1}^* are the gate edges of G_i. Eventually, the decomposition defines a fan graph G_r with gate edges e_r^* and e_{r+1}^* such that $e_{r+1}^* = \tilde{g}$.

We will show that each fan graph G_i with gate edges e_i^* and e_{i+1}^* admits a *block-drawing*. This is an outerplanar internally-strictly-convex grid drawing Γ_i of G_i satisfying the following properties:

B.1 The two gate edges e_i^* and e_{i+1}^* are drawn as vertical segments of unit length such that $y(u_i) = y(u_{i+1}) = 0$ and $y(v_i) = y(v_{i+1}) = 1$;

B.2 the vertices u_i and v_i are the leftmost vertices of Γ_i and no other vertex of G_i has the same x-coordinate as these vertices; and

B.3 the vertices u_{i+1} and v_{i+1} are the rightmost vertices of Γ_i and no other vertex of G_i has the same x-coordinate as these vertices.

We proceed as follows. Lemmas 9 and 10 are two technical lemmas that yield special internally-strictly-convex grid drawings of cycles. We use these two lemmas to compute a block-drawing Γ_i of each fan graph G_i with appropriate area bounds. We do so by drawing each of its faces using either Lemma 9 or Lemma 10, and we appropriately "merge" them together (Lemma 11). The final drawing Γ of G is obtained by "gluing" the block-drawings $\Gamma_0, \ldots, \Gamma_r$ (Theorem 12) at their shared gate edges.

In the following, by *half-plane of a line* we mean an open half-plane bounded by the line.

▶ **Lemma 9** (⋆). *Let $C = (v_1, v_2, \ldots, v_h)$ with $h \geq 3$ be a cycle. Let s_1 be a segment representing $e_1 = (v_1, v_2)$ such that v_1 is at the origin $(0, 0)$, $x(v_2)$ is a non-negative integer, and $y(v_2) = 1$. Then C admits an internally-strictly-convex grid drawing Γ_C such that C:*

I.1 *s_1 represents e_1 in Γ_C;*

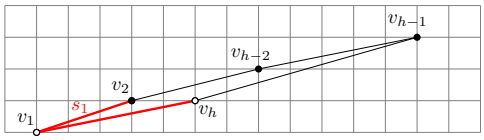


Figure 9 Illustrations for the proof of Lemma 9.

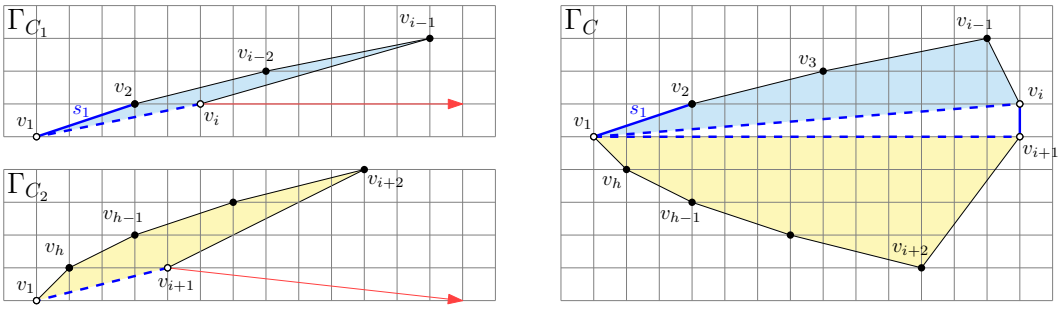


Figure 10 Illustration for the proof of Lemma 10.

I.2 *the vertex v_h is placed at $(x(v_2) + h - 2, 1)$;*

I.3 *let ℓ be the line that passes through s_1 oriented in the direction from v_1 to v_2. Then, all the vertices $v_3, \ldots, v_h$ lie in the right half-plane of ℓ;*

I.4 *the width of Γ_C is $x(v_2) \cdot (h-2) + \frac{(h-3)(h-2)}{2} + 1$; and*

I.5 *the height of Γ_C is $h - 1$.*

Sketch. The drawing Γ_C can be obtained as follows; see Fig. 9. Vertices v_1 and v_2 are at $(0, 0)$ and $(x(v_2), 1)$ in Γ_C. For $j = 3, \ldots, h-1$, we place v_j at $(x(v_2) \cdot (j-1) + \frac{(j-2)(j-1)}{2}, j-1)$. Finally, we place vertex v_h at $(x(v_2) + h - 2, 1)$. This completes the construction of Γ_C. We defer the proof that Γ_C satisfies Properties I.1–I.5 to the full version [3]. ◀

The following property is a consequence of the constraints of the drawings of Lemma 9.

▶ **Property 2.** *Let Γ_C be the drawing of a cycle $C = (v_1, v_2, \ldots, v_h)$ produced by Lemma 9. Then the drawing obtained by shifting vertex v_h in Γ_C by any integer amounts to the right and/or to the bottom is an internally-strictly-convex grid drawing of C.*

▶ **Lemma 10** (⋆). *Let $C = (v_1, v_2, \ldots, v_h)$ with $h > 3$ be a cycle. Let $e_1 = (v_1, v_2)$ and $e_2 = (v_i, v_{i+1})$ be two non-adjacent edges of C. Let s_1 be a segment that represents (v_1, v_2) such that v_1 is at the origin $(0, 0)$, $x(v_2)$ is a non-negative integer, and $y(v_2) = 1$. Then C admits an internally-strictly-convex grid drawing Γ_C such that:*

J.1 *s_1 represents e_1 in Γ_C;*

J.2 *the edge e_2 is represented by a vertical segment such that $x(v_i) > x(v_2)$, $y(v_i) = 1$, and $y(v_{i+1}) = 0$;*

J.3 *the width of Γ_C is $1 + \max\{x(v_2) \cdot (i-2) + \frac{(i-3)(i-2)}{2}, \frac{(h-i-1)(h-i)}{2}\}$; and*

J.4 *the height of Γ_C is $h - 2$.*

Sketch. We show how to construct the drawing Γ_C; refer to Fig. 10. First, we define two cycles $C_1 = (v_1, v_2, \ldots, v_i)$ and $C_2 = (v_1, v_h, v_{h-1}, \ldots, v_{i+1})$. We apply Lemma 9 to obtain a drawing Γ_{C_1} of C_1 in which the edge (v_1, v_2) is represented by s_1 and to obtain a drawing Γ_{C_2} of C_2 in which the edge (v_1, v_h) is represented by the segment connecting points $(0, 0)$

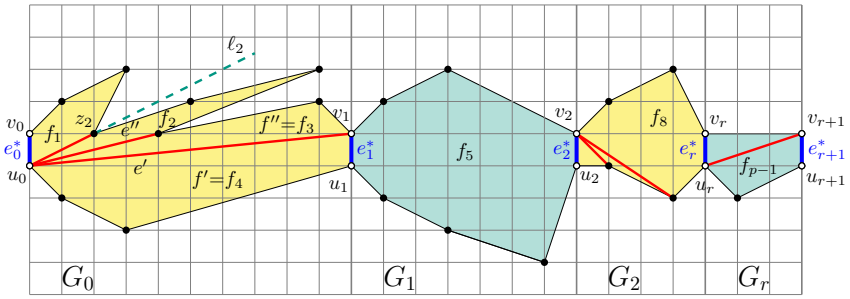


Figure 11 Illustration for Lemma 11 and Theorem 12. The drawing represents the graph in Fig. 8.

and $(1, 1)$. Let M be the maximum x-coordinate of a vertex in Γ_{C_1} and Γ_{C_2}. We leverage Property 2 to obtain a drawing Γ'_{C_1} of C_1 by shifting the vertex v_i horizontally and rightward in Γ_{C_1} so that its x-coordinate is $1 + M$. Analogously, we leverage Property 2 to obtain a drawing Γ'_{C_2} of C_2 by shifting the vertex v_{i+1} downward and rightward in Γ_{C_2} so that its x-coordinate is $1 + M$ and its y-coordinate is 0. To obtain the drawing Γ_C we then proceed as follows. We first initialize Γ_C to Γ'_{C_1}. Then we construct a vertically-mirrored copy Γ''_{C_2} of Γ'_{C_2}, that is, we flip the sign of the y-coordinate of each vertex, and we insert Γ''_{C_2} in Γ_C so that the placement of v_1 is the same in both drawings. Finally, we remove from Γ_C the drawing of the edges (v_1, v_i) and (v_1, v_{i+1}) and draw the edge (v_i, v_{i+1}) as a vertical segment of unit length. This concludes the construction of Γ_C. We defer the proof that Γ_C satisfies Properties J.1–J.4 to the full version [3]. ◀

▶ **Property 3.** *Let Γ_C be the drawing of a cycle $C = (v_1, v_2, \ldots, v_i, v_{i+1}, \ldots, v_h)$ produced by Lemma 10. Then the drawing obtained by shifting vertices v_i and v_{i+1} in Γ_C by the same integer amount to the right is an internally-strictly-convex grid drawing of C.*

We next show how to construct a block-drawing of a fan graph with appropriate area bounds.

▶ **Lemma 11** (⋆). *Every n_i-vertex fan graph G_i with internal faces of size at most k_i admits a block-drawing whose width is $O(n_i k_i)$ and whose height is $O(k_i)$.*

Sketch. We show how to construct a block drawing Γ_i of an n_i-vertex fan graph G_i with gate edges $e_i^* = (u_i, v_i)$ and $e_{i+1}^* = (u_{i+1}, v_{i+1})$ in the more interesting case in which one of u_i and v_i, say u_i, is incident to at least two internal edges (see, e.g., G_0 in Fig. 11). Let $g_1, g_2, \ldots, g_{t+2}$ be the internal faces of G_i in clockwise order around u_i, where g_1 is incident to e_i^*. Then, for $j = 1, \ldots, t$, we draw the cycle bounding face g_j by applying Lemma 9 with e_1 being e_i^* and s_1 being the segment between $(0, 0)$ and $(0, 1)$, if $j = 1$, and e_1 being the unique edge (u_i, z_j) shared by g_{j-1} and g_j, and s_1 being the segment representing (u_i, z_j) in the drawing constructed so far, otherwise. Finally, we draw the cycle C, obtained by merging the boundaries of g_{t+1} and g_{t+2} and by removing their shared edge (u_i, z_{t+2}), by applying Lemma 10 with e_1 being the edge (u_i, z_{t+1}), s_1 being the segment representing the edge (u_i, z_{t+1}), and e_2 being e_{i+1}^*. Then, by Property 3, we shift to the right vertices u_{i+1} and v_{i+1} so that they lie one unit to the right of the rightmost of the remaining vertices of G_i. Finally, we obtain the desired block drawing Γ_i of G_i by simply drawing the removed edge (u_i, z_{t+2}) as a straight-line chord of the strictly-convex polygon representing C. We defer the proof that Γ_C is a block drawing with the desired width and height bounds, as well as the remaining cases of the proof, to the full version [3]. ◀

We are now ready to prove the main theorem of this section.

▶ **Theorem 12.** *Every n-vertex outerpath whose internal faces have size at most k admits an internally-strictly-convex grid drawing in $O(nk^2)$ area.*

Proof. We compute block drawings $\Gamma_0, \ldots, \Gamma_r$ of the fan graphs $G_0, \ldots, G_r$ of the outerpath G so that each block drawing has height $O(k_i)$ and width $O(n_i k_i)$ by Lemma 11, where n_i is the number of vertices of G_i and k_i is the maximum size of any face of G_i $(0 \leq i \leq r)$. We iteratively construct Γ by "gluing" together the block drawings $\Gamma_0, \ldots, \Gamma_r$ at their common gate edges. More precisely, we proceed as follows. We initialize Γ to Γ_0. Then, for $i = 1, \ldots, r$, we augment Γ with a copy of Γ_i so that the drawing of the gate edge e_i^* in Γ_i coincides with the drawing of e_i^* in the drawing constructed so far. It follows that the height of Γ is $O(\max_{0 \leq i \leq r}\{k_i\})$, which is in $O(k)$, while its width is $O(\sum_{i=0}^{r} n_i k_i)$, which is in $O(nk)$. ◀

5 Conclusions

In this paper we have proved the first sub-quadratic area upper bound for internally-convex grid drawings of outerplanar graphs. We also presented algorithms to construct internally-strictly-convex grid drawings of outerpaths whose area is asymptotically optimal, with respect to the number of vertices of the graph and the maximum size of an internal face.

Several intriguing questions are left open by our research:

- A first, natural, research direction is to narrow the gap between the $O(n^{1.5})$ upper bound and the $\Omega(n)$ lower bound for internally-convex grid drawings of n-vertex outerplanar graphs. In particular, any super-linear lower bound would be an important step towards proving an analogous result for (not necessarily convex) planar straight-line grid drawings.
- Concerning internally-strictly-convex grid drawings of (general) n-vertex outerplane graphs whose faces have size at most k, for which we proved an $\Omega(nk^2)$ lower bound, we are not aware of any upper bound better than $O(n^3)$, which comes from the literature on strictly-convex polygons [1]. Tightening this gap is a nice goal.
- The case $k = 4$, for which there is an $O(n^2)$ upper bound (a consequence of results in [2]), is especially interesting. Indeed, any upper bound would translate to an upper bound for the area requirements of straight-line drawings of outer-1-planar graphs.
- Finally, we have observed that, for convex grid drawings of n-vertex outerplane graphs (in which the outer face is also required to be convex), $\Theta(n^3)$ is a tight bound for the area requirements. However, if the maximum size of an internal face is bounded, then our lower bound does not hold anymore. We believe that an $\Omega(n \log n)$ lower bound can be proved, given that the outer face would require a dimension to have $\Omega(n)$ length in order to be convex and given that there exist n-vertex trees that require $\Omega(\log n)$ length in both dimensions in any planar straight-line drawing (see, e.g., [14]). However, this lower bound is far from the $O(n^3)$ upper bound and it would be interesting to close this gap.

We conclude by mentioning that we believe that the techniques we developed in order to construct internally-strictly-convex grid drawings of outerpaths could lead to non-trivial area bounds for outerplanar graphs whose weak dual tree has bounded diameter. We plan to investigate the truth of this intuition in the near future.

References

1 G. E. Andrews. A lower bound for the volume of strictly convex bodies with many boundary lattice points. *Transactions of the American Mathematical Society*, 106:270–279, 1963.

2 Imre Bárány and Günter Rote. Strictly convex drawings of planar graphs. *Documenta Mathematica*, 11:369–391, 2006. doi:10.4171/DM/214.

3 Michael A. Bekos, Giordano Da Lozzo, Fabrizio Frati, Giuseppe Liotta, and Antonios Symvonis. Internally-convex drawings of outerplanar graphs in small area. *CoRR*, abs/2508.19913, 2025. URL: https://arxiv.org/abs/2508.19913.

4 Michael A. Bekos, Martin Gronemann, Fabrizio Montecchiani, and Antonios Symvonis. Strictly-convex drawings of 3-connected planar graphs. *J. Comput. Geom.*, 15(1):1–20, 2024. doi:10.20382/JOCG.V15I1A1.

5 Therese Biedl. Drawing outer-planar graphs in $O(n \log n)$ area. In Stephen G. Kobourov and Michael T. Goodrich, editors, *10th International Symposium on Graph Drawing (GD 2002)*, volume 2528 of *LNCS*, pages 54–65. Springer, 2002. doi:10.1007/3-540-36151-0_6.

6 Therese Biedl, Giuseppe Liotta, Jayson Lynch, and Fabrizio Montecchiani. Optimal-area visibility representations of outer-1-plane graphs. *J. Comput. Geom.*, 15(1):109–142, 2024. doi:10.20382/JOCG.V15I1A5.

7 Franz-Josef Brandenburg, David Eppstein, Michael T. Goodrich, Stephen G. Kobourov, Giuseppe Liotta, and Petra Mutzel. Selected open problems in graph drawing. In Giuseppe Liotta, editor, *11th International Symposium on Graph Drawing (GD 2003)*, volume 2912 of *LNCS*, pages 515–539. Springer, 2003. doi:10.1007/978-3-540-24595-7_55.

8 Timothy M. Chan. A near-linear area bound for drawing binary trees. *Algorithmica*, 34(1):1–13, 2002. doi:10.1007/S00453-002-0937-X.

9 Marek Chrobak and Goos Kant. Convex grid drawings of 3-connected planar graphs. *Int. J. Comput. Geom. Appl.*, 7(3):211–223, 1997. doi:10.1142/S0218195997000144.

10 Hubert de Fraysseix, János Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990. doi:10.1007/BF02122694.

11 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.

12 Giuseppe Di Battista and Fabrizio Frati. Small area drawings of outerplanar graphs. *Algorithmica*, 54(1):25–53, 2009. doi:10.1007/S00453-007-9117-3.

13 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.

14 Stefan Felsner, Giuseppe Liotta, and Stephen K. Wismath. Straight-line drawings on restricted integer grids in two and three dimensions. *J. Graph Algorithms Appl.*, 7(4):363–398, 2003. doi:10.7155/JGAA.00075.

15 Fabrizio Frati. Straight-line drawings of outerplanar graphs in $O(dn \log n)$ area. *Comput. Geom.*, 45(9):524–533, 2012. doi:10.1016/J.COMGEO.2010.03.007.

16 Fabrizio Frati, Maurizio Patrignani, and Vincenzo Roselli. LR-drawings of ordered rooted binary trees and near-linear area drawings of outerplanar graphs. *J. Comput. Syst. Sci.*, 107:28–53, 2020. doi:10.1016/J.JCSS.2019.08.001.

17 Ashim Garg and Adrian Rusu. Area-efficient planar straight-line drawings of outerplanar graphs. *Discret. Appl. Math.*, 155(9):1116–1140, 2007. doi:10.1016/J.DAM.2005.12.008.

18 Goos Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16(1):4–32, 1996. doi:10.1007/BF02086606.

19 Charles E. Leiserson. Area-efficient graph layouts (for VLSI). In *21st Annual Symposium on Foundations of Computer Science (FOCS 1980)*, pages 270–281. IEEE Computer Society, 1980. doi:10.1109/SFCS.1980.13.

20 Walter Schnyder. Embedding planar graphs on the grid. In *ACM-SIAM Symposium on Discrete Algorithms (SODA 1990)*, pages 138–148. SIAM, 1990. URL: http://dl.acm.org/citation.cfm?id=320176.320191.

Crossing and Non-Crossing Families

Todor Antić
Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Martin Balko
Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Birgit Vogtenhuber
Institute of Algorithms and Theory, Graz University of Technology, Austria

Abstract

For a finite set P of points in the plane in general position, a *crossing family* of size k in P is a collection of k line segments with endpoints in P that are pairwise crossing. It is a long-standing open problem to determine the largest size of a crossing family in any set of n points in the plane in general position. It is widely believed that this size should be linear in n.

Motivated by results from the theory of partitioning complete geometric graphs, we study a variant of this problem for point sets P that do not contain a *non-crossing family* of size m, which is a collection of 4 disjoint subsets P_1, P_2, P_3, and P_4 of P, each containing m points of P, such that for every choice of 4 points $p_i \in P_i$, the set $\{p_1, p_2, p_3, p_4\}$ is such that p_4 is in the interior of the triangle formed by p_1, p_2, p_3. We prove that, for every $m \in \mathbb{N}$, each set P of n points in the plane in general position contains either a crossing family of size $n/2^{O(\sqrt{\log m})}$ or a non-crossing family of size m, by this strengthening a recent breakthrough result by Pach, Rubin, and Tardos (2021). Our proof is constructive and we show that these families can be obtained in expected time $O(nm^{1+o(1)})$. We also prove that a crossing family of size $\Omega(n/m)$ or a non-crossing family of size m in P can be found in expected time $O(n)$.

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

Keywords and phrases crossing family, non-crossing family, geometric graph

Digital Object Identifier 10.4230/LIPIcs.GD.2025.19

Related Version *Full Version*: https://arxiv.org/abs/2508.17277

Funding *Todor Antić*: supported by grant no. 23-04949X of the Czech Science Foundation (GAČR). *Martin Balko*: supported by grant no. 23-04949X of the Czech Science Foundation (GAČR) and by the Center for Foundations of Modern Computer Science (Charles Univ. project UNCE 24/SCI/008).

Acknowledgements We would like to thank the organizers of the the 19th European Research Week on Geometric Graphs (GGWeek 2024) where this research was initiated. We would also like to thank Joachim Orthaber, Bettina Speckmann, and Viola Mészáros for interesting discussions about the problem during the early stages of the research. We also thank the anonymous reviewers for their valuable comments and suggestions, which helped improve the quality of this work.

1 Introduction

Let P be a finite set of points in the plane. We say that P is *in general position* if there is no line containing three points of P. Two line segments in the plane are *crossing* if their intersection is a single point in the relative interior of both of them. If $\mathcal{F}$ is a set of k line segments in the plane, each having both endpoints in P, then we say that $\mathcal{F}$ is a *crossing family of size k* in P if any two line segments in $\mathcal{F}$ cross; see Figure 1(a) for an example.



© Todor Antić, Martin Balko, and Birgit Vogtenhuber;
licensed under Creative Commons License CC-BY 4.0



A classic topic in discrete geometry is to find the largest size $T(n)$ of a crossing family in any set of n points in the plane in general position. The study of this problem was initiated in 1991 by Aronov, Erdős, Goddard, Kleitman, Klugerman, Pach, and Schulman [9], who proved that $T(n) \in \Omega(\sqrt{n})$. On the other hand, it is trivial to see that $T(n) \leq \lfloor n/2 \rfloor$. The currently best known upper bound is $T(n) \leq \lceil 8n/41 \rceil$, obtained by Aichholzer, Kynčl, Scheucher, Vogtenhuber, and Valtr [3]. There is a prevailing conjecture that $T(n) \in \Theta(n)$ [9, 14].

▶ **Problem 1** ([14]). *Is there a constant $c > 0$ such that every set of n points in the plane in general position contains a crossing family of size at least cn?*

Despite many attempts and several studied variants of the problem – see for example [3, 5, 19, 24, 29, 33] – to date no one has been able to resolve this problem. In fact, for a long time, not even an improvement of the lower bound of $\Omega(\sqrt{n})$ was obtained. In 2021, Pach, Rubin, and Tardos [29] achieved a breakthrough result by proving that $T(n) \in n^{1-o(1)}$. More precisely, they showed the following result.

▶ **Theorem 1** ([29]). *Every set of n points in the plane in general position contains a crossing family of size at least $n/2^{O(\sqrt{\log n})}$.*

Despite this remarkable progress, the problem of deciding whether $T(n) \in \Theta(n)$ remains open. Note that the problem becomes trivial if restricted to point sets in *convex position*, that is, point sets whose points are vertices of a convex polygon. It is easy to see that any set of n points in convex position determines a crossing family of size $\lfloor n/2 \rfloor$.

In this work, we study a variant of the crossing family problem where we relate the size of a largest crossing family in a point set to how convex/non-convex the point set is. A *non-crossing family* in a set P of points in the plane is a collection of four disjoint non-empty subsets P_1, P_2, P_3, and P_4 of P, such that for every choice of four points $p_i \in P_i$ the set $\{p_1, p_2, p_3, p_4\}$ is not in convex position, with p_4 in the interior of the convex hull of $\{p_1, p_2, p_3\}$. The *size* of a non-crossing family is the minimum over the cardinalities of the four subsets P_1, P_2, P_3, and P_4. see Figure 1(b) for an example.

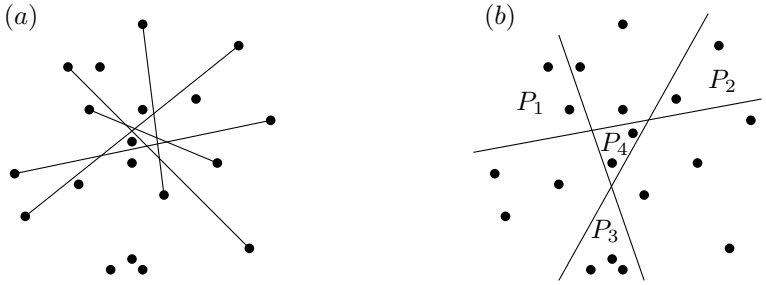


Figure 1 (a) A crossing family of size 5. (b) A non-crossing family of size 2.

It is easy to see and follows from Carathédory's theorem that a point set is in convex position if and only if it does not contain a non-crossing family of size one. Then, as we have mentioned, it is easy to find a linear-size crossing family. Towards answering the question of whether $T(n) \in \Theta(n)$, we parameterize point sets according to the largest non-crossing family they contain, and try to find their largest crossing family in dependence of this parameter. We thus study the following Ramsey-type problem.

▶ **Problem 2.** *Given positive integers m and n, what is the size of a largest crossing family in any set of n points in the plane in general position that does not contain a non-crossing family of size m?*

In particular, for which values of m can we always find a crossing family that is larger than the bound $n/2^{O(\sqrt{\log n})}$ from Theorem 1 for any set of n points in general position?

1.1 Motivation: partitioning of complete geometric graphs

The motivation for studying Problem 2 also comes from the theory of partitioning complete geometric graphs into subgraphs. A *complete geometric graph* is a pair (V, E), where V is a set of points in the plane in general position and E is the set of all closed line segments with endpoints in V. The elements of V are called *vertices* and the elements of E are called *edges*.

In 2006, Bose, Hurtado, Rivera-Campo, and Wood [12] introduced the problem of partitioning the edge set of a complete geometric graph into plane trees, or, more generally, into plane subgraphs. Recently, these problems have attracted considerable attention [4, 11, 17, 30].

It is not hard to see that the edge set of every complete geometric graph on n vertices can be partitioned into $n-1$ plane subgraphs. The best known lower bound $\frac{n}{2}+1$ [28] is also linear in n, but it is not known whether a partition into cn plane subgraphs for some constant $c<1$ is always possible. However, if every point set contains a large crossing family or a large non-crossing family, we get such an upper bound: Bose, Hurtado, Rivera-Campo, and Wood [12] proved that the edge set of every complete geometric graph on n vertices with k pairwise crossing edges can be partitioned into $n-k$ plane trees (which are not necessarily spanning). By Theorem 1, this gives an upper bound of $n - n/2^{O(\sqrt{\log n})}$ for the partition problem. On the other hand, Pach, Saghafian, and Schnider [30] showed that the edge set of every complete geometric graph on n vertices whose vertex set contains a non-crossing family of size bn can be partitioned into $(1-b)n$ plane subgraphs.

Motivated by these two results, Orthaber [27] posed the following problem.

▶ **Problem 3** ([27]). *Is there a constant $c>0$ such that every set of n points in the plane in general position contains a crossing family of size cn or a non-crossing family of size cn?*

An affirmative solution to Problem 3 would imply that the edge set of any complete geometric graph on n vertices can be partitioned into cn plane subgraphs for some constant $c<1$. This is a further motivation for studying Problem 2.

We note that a weaker variant of Problem 3 was independently posed by Schnider [32]; see Problem 5 in Section 6. Bose, Hurtado, Rivera-Campo, and Wood [12] actually proved a stronger result, where the crossing family of size k is replaced by a more general configuration called a *spoke set* by Schnider [32]. A *spoke set of size k* in a set P of n points in the plane in general position is a set $\mathcal{L}$ of k pairwise non-parallel lines such that in each unbounded region of the arrangement defined by the lines in $\mathcal{L}$ there lies at least one point of P; see Figure 2(a) for an example. It is not difficult to see that if P determines a crossing family of size k, then it determines a spoke set of size k [12]. The converse, however, is not true, as the example in Figure 2(b) illustrates.

Non-crossing families were also used by Dumitrescu and Pach [17], who proved that every complete geometric graph on a dense set of n points can be partitioned into at most cn plane subgraphs for some constant $c<1$. Here, a set P of n points is *dense* if the ratio between the maximum and the minimum distances in P is of the order of $\Theta(\sqrt{n})$.

1.2 Motivation: geometric quasiplanarity of complete graphs

For an integer $k \geq 2$, we say that a graph drawing is *k-quasiplanar* if it contains no set of k pairwise crossing edges. We say that a drawing of a graph G is *straight-line* if each edge of G is represented by a line segment between its endvertices. A graph G is *k-quasiplanar* if it admits a k-quasiplanar drawing. If G admits a straight-line k-quasiplanar drawing, then G is *geometric k-quasiplanar*.

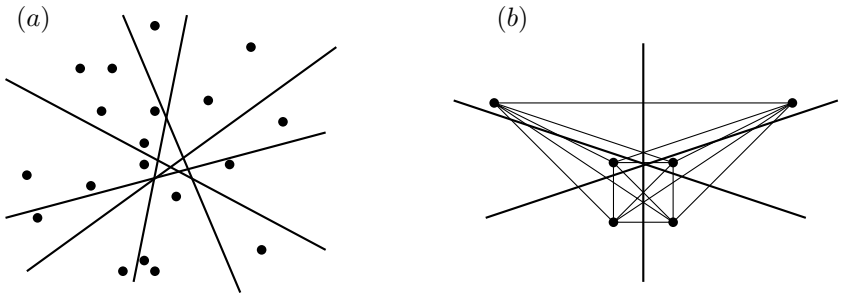


Figure 2 (a) A spoke set of size 5. (b) A set of 6 points in the plane in general position that determines a spoke set of size 3 but does not determine a crossing family of size 3.

The study of k-quasiplanar graphs was initiated in the 1990s and has gathered considerable attention from researchers in discrete geometry and graph drawing communities; see [1, 2, 6, 7, 8, 13, 16, 23, 22], among many others. One of the problems in this line of research is determining the minimum value k for which the complete graph K_n is k-quasiplanar or geometric k-quasiplanar. These values are only known for small values of n. In particular, Ackerman [1] showed that K_9 is geometric 3-quasiplanar and Brandenburg [13] proved that K_{10} is 3-quasiplanar but not geometric 3-quasiplanar. For larger values of n, it remains an interesting open problem to determine these values. In this language, Problem 2 asks us to determine this value for specific classes of straight-line drawings of K_n, that is, the ones for which the underlying point set does not contain a non-crossing family of size m.

1.3 Notation

In this paper, we consider only finite sets of points in the plane in general position. For brevity, we sometimes just refer to them as point sets. We use log to denote the logarithm with base 2. For a positive integer n, we denote by $[n]$ the set $\{1, \ldots, n\}$. For the sake of clarity of presentation, we systematically omit floor and ceiling signs whenever they are not crucial. For a point set P, we use $\mathrm{conv}(P)$ to denote the convex hull of P. If A and B are two sets of points in the plane, then we call them *separated* if their convex hulls are disjoint.

2 Our contribution

We first show that forbidding non-crossing families of some small size can indeed help in finding larger crossing families than the ones obtained from Theorem 1. We actually find something stronger, a so-called *convex bundle*.

For a positive integer k and a point set P in the plane, a *convex bundle of size k* in P is a k-tuple of disjoint non-empty subsets $C_1, \ldots, C_k$ of P, such that for every choice of points $c_i \in C_i$ for every $i \in [k]$, the points $c_1, \ldots, c_k$ are in convex position. The *width* of a convex bundle of size k is the minimum over the cardinalities of its subsets $C_1, \ldots, C_k$.

Our first result guarantees the existence of a convex bundle of size $\Theta(n/m)$ in any n-point set without a non-crossing family of size m. To show it, we make use of the so-called Same-Type-Lemma [10], a classic result on orientations of point tuples in point sets.

▶ **Theorem 2.** *There is a constant $C > 0$ such that, for all positive integers m and k, every set of at least Ckm points in the plane in general position contains either a convex bundle of size k and width m or a non-crossing family of size m.*

Since every convex bundle of size $2k$ and width at least 1 determines a crossing family of size k, we obtain the following immediate corollary of Theorem 2.

▶ **Corollary 3.** *There is a constant $C > 0$ such that, for all positive integers m and k, every set of at least Ckm points in the plane in general position contains either a crossing family of size k or a non-crossing family of size m.* ◀

In particular, it follows from Corollary 3 that if our point set does not contain a constant size non-crossing family, then it contains a crossing family of linear size. Working with crossing families directly, we use Theorem 2 in combination with Theorem 5 below to improve the bound from Corollary 3 as follows, by this obtaining our main theorem.

▶ **Theorem 4.** *There is a constant $C' > 0$ such that, for every positive integer m, every set of $n \geq C'm$ points in the plane in general position contains either a non-crossing family of size m or a crossing family of size $n/2^{O(\sqrt{\log m})}$.*

Since no set of n points in the plane contains a non-crossing family of size n, Theorem 1 can be obtained as a corollary of Theorem 4 by setting $m = n$. The lower bound on the size of the crossing families that we obtain by Theorem 4 is asymptotically larger than the bounds from Theorem 1 and Corollary 3, as long as the point set does not contain a non-crossing family of size $n^{\Omega(1)}$. For example, if we forbid non-crossing families of size $\log n$, then we find a crossing family of size $\Omega(n/2^{\sqrt{\log\log n}})$ instead of $n/2^{O(\sqrt{\log n})}$ and $\Omega(n/\log n)$ that we would get from Theorem 1 and Corollary 3, respectively. Further, Theorem 4 implies that for any constant m, if P is a set of n points in the plane in general position with no non-crossing family of size m, then the complete geometric graph on P can be decomposed into cn plane subgraphs for some constant $c < 1$, by this resolving the problem of Bose, Hurtado, Rivera-Campo and Wood [12] for such point sets.

To prove Theorem 4, we use the following slight strengthening of Theorem 1, which we obtain by modifying the proof of the result by Pach, Rubin, and Tardos [29].

▶ **Theorem 5.** *For every set P of $n \geq 2$ points in the plane in general position and every partition of P into two separated subsets P_1 and P_2 such that $||P_1| - |P_2|| \leq 1$, P contains a crossing family $\mathcal{F}$ of size at least $n/2^{O(\sqrt{\log n})}$ where each segment from $\mathcal{F}$ has one endpoint in P_1 and one endpoint in P_2.*

We also consider the algorithmic aspects of our results. The proofs of Theorems 2 and 4 are both constructive. By analyzing their time complexity, we obtain an expected linear time algorithm for Theorem 2 and an expected polynomial time algorithm for Theorem 4.

▶ **Theorem 6.** *There is a constant $C > 0$ such that for all positive integers k and m, if P is a set of $n = Ckm$ points in the plane in general position, then a convex bundle of size k and width m or a non-crossing family of size m can be computed in expected time $O(n)$.*

Further, a crossing family of size $n/2^{O(\sqrt{\log m})}$ or a non-crossing family of size m can be computed in expected time $O(nm^{1+O((\log m)^{-1/3})})$.

We remark that the non-determinism in the time complexities stems from an analysis of the runtime requirements of the Same-Type-Lemma. A deterministic variant of this building block would yield a deterministic analogue of Theorem 6. Further, we note that the proof of Theorem 1 is also constructive and results in an $n^{2+O((\log n)^{-1/3})}$ time algorithm for finding an according crossing family in a set of n points; see [29].

Finally, we show that the size of the largest spoke set and the largest crossing family in a point set can differ by a constant multiplicative factor, showcasing the different behavior of these two closely related notions.

▶ **Proposition 7.** *For any odd integer k, there is a set P of $3k-1$ points in the plane in general position, such that the largest spoke set in P contains $\lfloor 1.5k \rfloor$ lines but P does not contain a crossing family of size larger than k.*

Note that if Problem 1 admits an affirmative solution, then the sizes of the maximal spoke set and the maximal crossing family differ by at most a constant multiplicative factor.

3 Proofs of Theorems 2 and 4

This section is devoted to proving our main result, the existence of a large crossing family or a large non-crossing family (Theorem 4), for which we will use and first prove the existence of a large convex bundle or a large non-crossing family (Theorem 2).

One crucial ingredient for the proof of Theorem 2 is a well-known result by Bárány and Valtr [10], called the *Same-Type lemma*, on the combinatorics of point sets in $\mathbb{R}^d$.

For an integer $d \geq 2$ and non-coplanar points $p_1, \ldots, p_{d+1} \in \mathbb{R}^d$, where $p_{i,j}$ is the jth coordinate of p_i, the *orientation* of the $(d+1)$-tuple $(p_1, \ldots, p_{d+1})$ is the sign of the determinant of the matrix

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ p_{1,1} & p_{2,1} & \cdots & p_{d+1,1} \\ \vdots & \vdots & \vdots & \vdots \\ p_{1,d} & p_{2,d} & \cdots & p_{d+1,d} \end{pmatrix}.$$

For point sets $Y_1, \ldots, Y_{d+1}$ in $\mathbb{R}^d$, we say that the sets $Y_1, \ldots, Y_{d+1}$ have the *same-type property* if, for every choice of points $y_1 \in Y_1, \ldots, y_{d+1} \in Y_{d+1}$, the orientation of the $(d+1)$-tuple $(y_1, \ldots, y_{d+1})$ is the same. More generally, for every integer $r \geq d+1$, we say that sets $Y_1, \ldots, Y_r$ in $\mathbb{R}^d$ have the *same-type property* if any $d+1$ of them do.

With these definitions, we can now state the Same-Type lemma.

▶ **Theorem 8** (Same-Type lemma [10])**.** *For all integers $d \geq 2$ and $r \geq d+1$, there is a constant $c = c(d,r) > 0$ such that for all pairwise disjoint sets $X_1, \ldots, X_r$ in $\mathbb{R}^d$ whose union is in general position, there exist point sets $Y_1, \ldots, Y_r$ having the same-type property and satisfying $Y_i \subseteq X_i$ and $|Y_i| \geq c|X_i|$.*

Bárány and Valtr [10] showed that $c(d,r) \geq (d+1)^{(-2^d-1)\binom{r-1}{d}}$. This was later improved by Fox, Pach, and Suk [18] to $c(d,r) \in 2^{-O(d^3 r \log r)}$. Very recently, Bukh [15] proved the bounds $d^{-50d^3} r^{-d^2} \leq c(d,r) \leq d^d r^{-d}$, which are polynomial in r.

For disjoint sets P and Q of points in the plane, we write $x(P) < x(Q)$ if the x-coordinate $x(p)$ of every point $p \in P$ is smaller than $x(q)$ for every point $q \in Q$. For a positive integer a, we say that a points in the plane form an *a-cap* if they lie on the graph of some concave function. Similarly, for $u \in \mathbb{N}$, a set of u points in the plane forms a *u-cup* if its points lie on the graph of some convex function. Note that any a-cap or u-cup is a point set in convex position.

▶ **Theorem 2.** *There is a constant $C > 0$ such that, for all positive integers m and k, every set of at least Ckm points in the plane in general position contains either a convex bundle of size k and width m or a non-crossing family of size m.*

Proof. We set $C = 7 \cdot c(2,7)^{-1} \cdot c(2,5)^{-6}$, where $c(2,r) \in (0,1]$ is the constant from Theorem 8. We assume $k \geq 3$ as otherwise the statement is trivial. Let P be a set of $n = Ckm$ points in the plane in general position that does not contain a non-crossing family of size m. We

partition P by vertical lines into sets $P_1, \dots, P_7$ such that $x(P_1) < \dots < x(P_7)$ and $|P_i| = n/7$ for every $i \in \{1, \dots, 7\}$. We now apply Theorem 8 to the 7-tuple $(P_1, \dots, P_7)$ and obtain the 7-tuple $\mathcal{B} = (P'_1, \dots, P'_7)$ that satisfies $P'_i \subseteq P_i$ and $|P'_i| \geq c(2,7) \cdot n/7 \geq m$ for every $i \in \{1, \dots, 7\}$.

We let p_i be an arbitrary point from P'_i for each $i \in \{1, \dots, 7\}$. We show that the set $S = \{p_1, \dots, p_7\}$ is in convex position. Suppose for contradiction it is not. Then, by Carathéodory's theorem, there is a 4-tuple T of points from S such that T is not in convex position. Since the collection $\mathcal{B}$ has the same-type property, the sets from $\mathcal{B}$ whose points are in T form a non-crossing family of size m. This contradicts our assumption that P does not contain such a family. Moreover, since $x(P'_1) < \dots < x(P'_7)$, it follows from Theorem 8 that there are sets A and U such that $A \cup U = \{1, \dots, 7\}$, $A \cap U = \{1, 7\}$, $\{p_i : i \in A\}$ is an $|A|$-cap, and $\{p_i : i \in U\}$ is a $|U|$-cup for every choice of $S = \{p_1, \dots, p_7\}$. By the pigeonhole principle, we have $|A| \geq 5$ or $|U| \geq 5$. By symmetry, we can assume without loss of generality that $|A| \geq 5$ and we let $a_1 < \dots < a_5$ be the 5 smallest elements of A.

Thus, we are done if $k \leq 5$, and we can assume $k \geq 6$. We partition the set P'_{a_3} by vertical strips into sets $P'_{a_3,1}, \dots, P'_{a_3,k}$, each of size $|P'_{a_3}|/k$. Now, we let $Q_1 = P'_{a_1}$, $Q_2 = P'_{a_2}$, $Q_{k-1} = P'_{a_4}$, $Q_k = P'_{a_5}$, and $Q_j = P'_{3,j}$ for every $j \in \{3, 4, \dots, k-2\}$. Observe that $x(Q_1) < \dots < x(Q_k)$ and $|Q_i| \geq c(2,7) \cdot n/(7k) \geq m$ for every $i \in [k]$. Moreover, for every choice of $q_1 \in Q_1$, $q_2 \in Q_2$, $q_3 \in Q_3 \cup \dots \cup Q_{k-2}$, $q_4 \in Q_{k-1}$, and $q_5 \in Q_k$, the points $q_1, \dots, q_5$ form a 5-cap.

Let π be the permutation of $[k]$ defined by $\pi(i) = \lceil i/2 \rceil$ if i is odd and $\pi(i) = k + 1 - i/2$ if i is even. Our goal is to show that for any $5 \leq \ell \leq k$, there exist large sets $R_i^\ell \subseteq Q_{\pi(i)}$ for $i \in [\ell]$ such that for all choices $r_i \in R_i^\ell$, the points $r_1, \dots, r_\ell$ form an ℓ-cap; cf. Figure 3. To this end, we first iteratively construct the sets R_i^ℓ such that any five sets $R_i^\ell, \dots, R_{i+4}^\ell$ fulfill the same-type property. Then, we reason about their sizes, and finally, we show by induction that they fulfill ℓ-cap property.

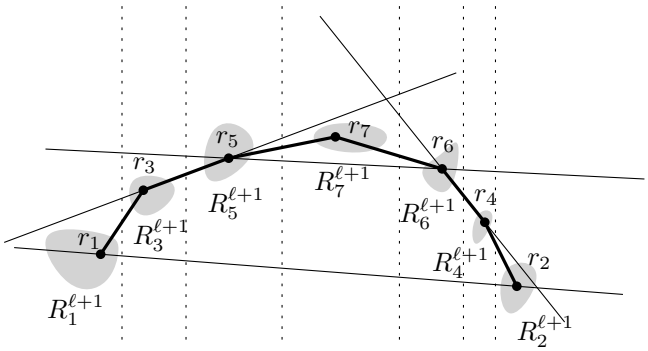


Figure 3 An illustration of the proof of Theorem 2 for $\ell = 6$.

For $\ell = 5$, we simply set $R_i^\ell = Q_{\pi(i)}$ for each $i \in [\ell]$. Note that these sets have the same-type property by construction, and that for all choices $r_i \in R_i^5$, the points $r_1, \dots, r_5$ form a 5-cap. We proceed by iteratively increasing ℓ. Assume that we have sets $R_1^\ell, \dots, R_\ell^\ell$ for some integer ℓ with $5 \leq \ell < k$ such that $R_i^\ell \subseteq Q_{\pi(i)}$ for every $i \in [\ell]$ and such that any five sets $R_i^\ell, \dots, R_{i+4}^\ell$ fulfill the same-type property. We apply Theorem 8 to the 5-tuple $R_{\ell-3}^\ell, R_{\ell-2}^\ell, R_{\ell-1}^\ell, R_\ell^\ell, R_{\ell+1}^\ell = Q_{\pi(\ell+1)}$ ordered by x-coordinates. This way, we obtain sets $R_j^{\ell+1} \subseteq R_j^\ell$ for every $j \in \{\ell-3, \dots, \ell+1\}$ such that $|R_j^{\ell+1}| \geq c(2,5) \cdot |R_j^\ell|$ and the collection $R_{\ell-3}^{\ell+1}, R_{\ell-2}^{\ell+1}, R_{\ell-1}^{\ell+1}, R_\ell^{\ell+1}, R_{\ell+1}^{\ell+1}$ has the same-type property. For every $i \in [\ell-4]$, we set $R_i^{\ell+1} = R_i^\ell$.

Next, consider the sizes of the final sets R_i^k for $i \in [k]$. Note that we apply Theorem 8 at most five times within each set Q_i. Thus, at the end we have, for every $i \in [k]$,

$$|R_i^k| \geq c(2,5)^5 \cdot |Q_{\pi(i)}| \geq c(2,5)^5 \cdot c(2,7) \cdot n/(7k) \geq m/c(2,5),$$

where we used the choice of C and n.

Finally, we argue by induction on ℓ that for all choices $r_i \in R_i^\ell$, the points $r_1, \ldots, r_\ell$ form an ℓ-cap. This is clearly true for $\ell = 5$ by the choice of Q_1, Q_2, Q_3, Q_{k-1}, and Q_k. Thus, we assume $\ell \geq 5$ and proceed with the induction step $\ell \to \ell + 1$.

Since $R_j^{\ell+1} \subseteq R_j^\ell$ for every $j \in [\ell]$, it follows from the induction hypothesis that the points $r_1, \ldots, r_\ell$ form an ℓ-cap for any choice of $r_i \in R_i^{\ell+1}$ with $i \in [\ell]$. Thus, we only need to verify that $r_{\ell+1}$ extends this ℓ-cap into an $(\ell+1)$-cap.

Since $x(R_{\ell-1}^{\ell+1}) < x(R_{\ell+1}^{\ell+1}) < x(R_\ell^{\ell+1})$ or $x(R_\ell^{\ell+1}) < x(R_{\ell+1}^{\ell+1}) < x(R_{\ell-1}^{\ell+1})$, all points from $R_{\ell+1}^{\ell+1}$ lie in a vertical strip S between $R_{\ell-1}^{\ell+1}$ and $R_\ell^{\ell+1}$. Since for every choice of $q_1 \in Q_1$, $q_2 \in Q_2$, $q_3 \in Q_3 \cup \cdots \cup Q_{k-2}$, $q_4 \in Q_{k-1}$, and $q_5 \in Q_k$, the points $q_1, \ldots, q_5$ form a 5-cap, all points from $\cup_{i \in \{3,4,\ldots,\ell+1\}} R_i^{\ell+1}$ lie above any line determined by one point from $R_1^{\ell+1}$ and one point from $R_2^{\ell+1}$.

Since the collection $R_{\ell-3}^{\ell+1}, R_{\ell-2}^{\ell+1}, R_{\ell-1}^{\ell+1}, R_\ell^{\ell+1}, R_{\ell+1}^{\ell+1}$ has the same-type property, for every choice of one point from each set of this collection, the resulting 5-tuple is in convex position. Otherwise, similarly as before, it follows from Carathéodory's theorem that P contains a non-crossing family of size $m/c(2,5) \geq m$, which is impossible by assumption. It follows from this fact that all points from $R_{\ell+1}^{\ell+1}$ lie below every line determined by a point from $R_{\ell-3}^{\ell+1}$ and a point from $R_{\ell-1}^{\ell+1}$ (otherwise there are four points from $R_{\ell-3}^{\ell+1}$, $R_{\ell-1}^{\ell+1}$, $R_{\ell+1}^{\ell+1}$, and $R_\ell^{\ell+1}$ that are not in convex position). Similarly, all points from $R_{\ell+1}^{\ell+1}$ lie below every line determined by a point from $R_{\ell-2}^{\ell+1}$ and a point from $R_\ell^{\ell+1}$. All points from $R_{\ell+1}^{\ell+1}$ lie on the same side of all lines determined by a point from $R_{\ell-1}^{\ell+1}$ and a point from $R_\ell^{\ell+1}$. If they lie above all such lines, then $r_1, \ldots, r_{\ell+1}$ form an $(\ell+1)$-cap, since $r_{\ell+1} \in S$ is contained in the triangle spanned by lines $\overline{r_{\ell-3}r_{\ell-1}}$, $\overline{r_{\ell-2}r_\ell}$, and $\overline{r_{\ell-1}r_\ell}$, and we are done. Thus, we suppose that all points from $R_{\ell+1}^{\ell+1}$ lie below these lines.

Then, for every choice $r_1 \in R_1^{\ell+1}$, $r_2 \in R_2^{\ell+1}$, $r_{\ell-1} \in R_{\ell-1}^{\ell+1}$, $r_\ell \in R_\ell^{\ell+1}$, and $r_{\ell+1} \in R_{\ell+1}^{\ell+1}$, the points $r_1, r_2, r_{\ell-1}, r_\ell, r_{\ell+1}$ are not in convex position. This is because $r_{\ell+1}$ is located in the part of the vertical strip S that lies between the two line segments r_1r_2 and $r_{\ell-1}r_\ell$. We apply Theorem 8 to the 5-tuple $(R_1^{\ell+1}, R_2^{\ell+1}, R_{\ell-1}^{\ell+1}, R_\ell^{\ell+1}, R_{\ell+1}^{\ell+1})$ and obtain sets $S_j \subseteq R_j^{\ell+1}$ for every $j \in \{1, 2, \ell-1, \ell, \ell+1\}$ such that $|S_j| \geq c(2,5) \cdot |R_j^{\ell+1}|$ and $(S_1, S_2, S_{\ell-1}, S_\ell, S_{\ell+1})$ has the same-type property. By selecting a point from each S_j we cannot get a point set in convex position, since no 5-tuple of points from $R_1^{\ell+1}, R_2^{\ell+1}, R_{\ell-1}^{\ell+1}, R_\ell^{\ell+1}, R_{\ell+1}^{\ell+1}$ is in convex position. It follows from the same-type property and from Carathéodory's theorem that some 4-tuple of sets S_j forms a non-crossing family. This family has size $|S_j| \geq c(2,5) \cdot |R_j^{\ell+1}| \geq m$, which contradicts the fact that P does not contain such a non-crossing family.

Altogether, the sets $R_1^k, \ldots, R_k^k$ form a convex bundle of size k and width at least m. ◀

To obtain the bound in Theorem 4, we combine Theorem 2 with Theorem 5 (the proof of which can be found in Section 4).

▶ **Theorem 4.** *There is a constant $C' > 0$ such that, for every positive integer m, every set of $n \geq C'm$ points in the plane in general position contains either a non-crossing family of size m or a crossing family of size $n/2^{O(\sqrt{\log m})}$.*

Proof. Let $C' = 6C$, where C is the constant from Theorem 2, and let P be a set of $n \geq C'm$ points in general position that does not contain a non-crossing family of size m. Further, let $k = \lfloor n/(2Cm) \rfloor \geq 3$, and note that $2kCm \leq n < 2(k+1)Cm \leq \frac{8}{3}kCm$. It follows from

Theorem 2 applied with $2k$ that P contains a convex bundle of size $2k$ and width m. We denote the sets which form this convex bundle by $A_1, A_2, \ldots, A_k, B_1, B_2, \ldots, B_k$ in cyclical order given by the convex position.

For every $i \in [k]$, we apply Theorem 5 to the set $A_i \cup B_i$ and obtain a crossing family $\mathcal{F}_i$ of size at least $m/2^{c\sqrt{\log m}}$ for some constant $c > 0$ such that each of its line segments has one endpoint in A_i and the other in B_i, where the constant c does not depend on the cardinality m of the point sets.

By the definition of the convex bundle, each line segment from $\mathcal{F}_i$ crosses each line segment from $\mathcal{F}_j$ for all i and j with $1 \leq i < j \leq k$. It follows that the line segments in $\cup_{i=1}^{k} \mathcal{F}_i$ form a crossing family of size

$$\frac{km}{2^{c\sqrt{\log m}}} \geq \frac{3}{8C} \cdot \frac{n}{2^{c\sqrt{\log m}}} \in \frac{n}{2^{O(\sqrt{\log m})}},$$

which completes the proof. ◀

4 Proof of Theorem 5

In this section, we show that every set of n points in the plane in general position partitioned into two separated nonempty subsets P_1 and P_2 of equal size contains a crossing family $\mathcal{F}$ of size at least $n/2^{O(\sqrt{\log n})}$, where each segment from $\mathcal{F}$ has one endpoint in P_1 and one endpoint in P_2. We do so by slightly modifying the proof of Theorem 1 by Pach, Rubin, and Tardos [29].

Let $\mathcal{L}$ be a finite set of lines in the plane. Following the terminology in [29], we call the partition $\mathcal{A} = \mathcal{A}(\mathcal{L})$ of $\mathbb{R}^2 \setminus (\bigcup \mathcal{L})$ into 2-dimensional sets, called *cells*, the *arrangement*[1] of lines from $\mathcal{L}$. Each cell of $\mathcal{A}$ is a maximal connected region of $\mathbb{R}^2 \setminus (\bigcup \mathcal{L})$ and a (possibly unbounded) convex polygon. The boundary of each cell consists of *edges*, which are portions of the lines from $\mathcal{L}$. The edges connect crossings between lines from $\mathcal{L}$, which are called *vertices*. The *zone* of a line $\ell \notin \mathcal{L}$ in the arrangement $\mathcal{A}$ is the union of all the open cells whose closure is intersected by ℓ.

We will need the following result on point sets and line arrangements, which was implicitly established by Matoušek [26, 25]; see [29] for a proof.

▶ **Lemma 9** ([26]). *For any n-element point set P in the plane in general position and for any $\varepsilon > 0$, there exists a set $\mathcal{L}$ of $O(1/\varepsilon^2)$ lines such that the zone of any line $\ell \notin \mathcal{L}$ within the arrangement $\mathcal{A}(\mathcal{L})$ contains at most εn points of P.*

The following definitions were introduced by Pach, Rubin, and Tardos [29]. For non-empty separated point sets A and B, we define a binary relation $<_B$ on A such that, for distinct points $x, y \in A$, we let $x <_B y$ if B is contained in the open half-plane to the left of the line determined by x and y and oriented from x towards y. Pach, Rubin, and Tardos [29] proved that $<_B$ is a partial order on A, in which two distinct points x and y are incomparable if and only if the line through x and y intersects $\mathrm{conv}(B)$.

For a partially ordered set $(S, <_S)$, we use $\iota(S, <_S)$ to denote the number of pairs of elements from S that are incomparable in $<_S$. For $\varepsilon > 0$, two separated point sets A and B, each with m points, are said to form an *ε-avoiding* pair if $\iota(A, <_B) + \iota(B, <_A) \leq \varepsilon m^2$.

[1] We remark that in other literature, the line arrangement formed by a set $\mathcal{L}$ of lines also contains $\mathcal{L}$, partitioned into vertices, which are the crossings of $\mathcal{L}$, and edges, which are the (possibly unbounded) segments of lines of $\mathcal{L}$ between consecutive crossings.

The following lemma is a variant of Lemma 3.3 by Pach, Rubin, and Tardos [29]. Its proof is a modification of their argument.

▶ **Lemma 10.** *There is an absolute constant $c > 0$ with the following property. For every positive integer m and every real number $\varepsilon \in (0,1)$, if P is a set of n points in the plane in general position such that $n \geq \frac{cm}{\varepsilon^4}$ and P is partitioned into two separated subsets P_1 and P_2 with $||P_1| - |P_2|| \leq 1$, then there are two (separated) m-element sets $A \subseteq P_1$ and $B \subseteq P_2$ that form an ε-avoiding pair.*

Proof. We first apply Lemma 9 to obtain a set $\mathcal{L}$ of $r = c_0/\varepsilon^2$ lines for some constant $c_0 > 0$ and an arrangement $\mathcal{A} = \mathcal{A}(\mathcal{L})$ such that the zone of any other line within $\mathcal{A}$ contains at most $\varepsilon n/16$ points of P. We assume that $r \geq 3$ and choose a constant c such that $c \geq 16c_0^2$.

We split each cell of $\mathcal{A}$ with parallel segments or half-lines that do not pass through any point of P into smaller cells such that all but at most one of them (which we will call *exceptional*) contain precisely m points of P, and if there is an exceptional cell, it contains fewer than m points. Since there is at most one exceptional (smaller) cell per cell of $\mathcal{A}$, there are at most $\frac{r^2+r+1}{2} < r^2$ exceptional cells in total. Let Π be the resulting cell decomposition of $\mathbb{R}^2$ induced by $\mathcal{A}$ and the additional segments and half-lines. Note that every cell in Π is convex. We call the set of m points of P inside a non-exceptional cell of Π a *cluster*. Let D be the number of clusters and note that $D \leq n/m$.

Let ℓ be a line that separates P_1 from P_2. Such a line exists as P_1 and P_2 are separated. By the choice of $\mathcal{A}$, the zone of ℓ in $\mathcal{A}$ contains at most $\varepsilon n/16$ points of P. These points lie in at most $\varepsilon n/(16m)$ clusters, as each such cluster contains m points from the zone of ℓ in $\mathcal{A}$.

There are at most $r^2 m$ points of P that do not belong to any cluster as the number of exceptional (smaller) cells is at most r^2 and each such a cell contains less than m points of P. Thus, since $|P_1| = n/2$, the number of clusters that contain points from P_1 is at least $(n/2 - r^2 m)/m$, and an analogous claim is true for P_2.

The number of pairs (C_1, C_2) of clusters C_1 and C_2 such that $C_1 \subseteq P_1$ and $C_2 \subseteq P_2$ is at least

$$\left(\frac{n/2 - r^2 m}{m}\right)^2 - \frac{\varepsilon n}{16m} D \geq \frac{(n/2 - c_0^2 m/\varepsilon^4)^2}{m^2} - \frac{\varepsilon n^2}{16m^2}$$
$$\geq \frac{(n/2 - n/16)^2}{m^2} - \frac{n^2}{16m^2} > \frac{n^2}{8m^2},$$

as every cluster containing points from P_1 and P_2 is in the zone of ℓ and the number of exceptional cells is at most r^2. We also used the choice of n, $r = c_0/\varepsilon^2$, $\varepsilon < 1$, $D \leq n/m$, $n \geq cm/\varepsilon^4$, and $c \geq 16c_0^2$.

Let

$$X = \sum_{(C_1, C_2)} (\iota(C_1, <_{C_2}) + \iota(C_2, <_{C_1})),$$

where the sum is taken over all ordered pairs (C_1, C_2) of distinct clusters such that $C_1 \subseteq P_1$ and $C_2 \subseteq P_2$.

We now estimate X according to the point pairs that are incomparable with respect to a cluster. An unordered pair of distinct points $\{x, y\} \in \binom{P}{2}$ contributes to X only if x and y come from the same cluster whose points from P are contained in P_1 or P_2. In this case, the contribution of $\{x, y\}$ is the number of other clusters whose points from P are contained in P_2 or P_1, respectively, and whose convex hulls are crossed by the line ℓ' containing x and y. All of these clusters belong to the zone of ℓ' in the arrangement $\mathcal{A}$. By the choice of $\mathcal{A}$,

the zone of any line contains at most $\varepsilon n/16$ vertices. Thus, the contribution of any pair of points is at most $\varepsilon n/(16m)$ and we obtain

$$X \leq D\binom{m}{2}\frac{\varepsilon n}{16m} < \frac{\varepsilon n^2}{32},$$

since $D \leq n/m$.

On the other hand, every pair of clusters (C_1, C_2) which is not ε-avoiding contributes more than εm^2 to X. Therefore, there are at most $X/(\varepsilon m^2)$ such pairs, implying that the number of not ε-avoiding pairs of clusters is less than

$$\frac{\varepsilon n^2}{32} \cdot \frac{1}{\varepsilon m^2} = \frac{n^2}{32m^2}.$$

Clearly, each cluster has m points of P and any two of them are separated.

Putting everything together, the number of pairs (C_1, C_2) of clusters such that $C_1 \subseteq P_1$ and $C_2 \subseteq P_2$ is at least $\frac{n^2}{8m^2}$, which is larger than the number $\frac{n^2}{32m^2}$ of pairs of clusters that are not ε-avoiding. Thus, there is a pair of clusters (A, B) that is ε-avoiding and satisfies $A \subseteq P_1$ and $B \subseteq P_2$. ◀

Finally, we state the following lemma by Pach, Rubin, and Tardos [29].

▶ **Lemma 11** ([29]). *Let s be a positive integer and set $K = 8^{\binom{s}{2}}$, $M = 9^s K$, $\varepsilon = 2^{-3s-11}$. Assume that A and B are M-element point sets that form an ε-avoiding pair and $A \cup B$ is in general position. Then, we can find K pairwise crossing segments, each connecting a point of A to a point of B.*

We are now ready to prove Theorem 5 by slightly modifying the proof of Theorem 1 by Pach, Rubin, and Tardos [29].

▶ **Theorem 5.** *For every set P of $n \geq 2$ points in the plane in general position and every partition of P into two separated subsets P_1 and P_2 such that $||P_1| - |P_2|| \leq 1$, P contains a crossing family $\mathcal{F}$ of size at least $n/2^{O(\sqrt{\log n})}$ where each segment from $\mathcal{F}$ has one endpoint in P_1 and one endpoint in P_2.*

Proof. Assuming $n \geq 3$, let s be the smallest positive integer such that P does not determine a crossing family of size $K = 8^{\binom{s}{2}}$ that has all line segments between P_1 and P_2. We have $s \geq 1$, since P determines a crossing family of size $1 = 8^{\binom{1}{2}}$.

By Lemma 11, there is no pair $\{A, B\}$ of sets, with A containing $M = 9^s K$ points of P_1 and B containing M points of P_2, that is $\varepsilon = 2^{-3s-11}$-avoiding. Applying Lemma 10 to P gives $n \in O(M\varepsilon^{-4}) = 2^{O(s)}K$.

By the choice of s, the point set P determines a crossing family of size $K' = 8^{\binom{s-1}{2}} \in K/2^{O(s)}$ with all line segments between P_1 and P_2. Therefore, we have $n > K' \in K/2^{O(s)}$ and $s = O(\sqrt{\log n})$. We also have $K' \in K/2^{O(s)} = n/2^{O(s)} = n/2^{O(\sqrt{\log n})}$. ◀

5 Proof of Theorem 6

This section is devoted to the algorithmic aspects of Theorems 2 and 4. Specifically, we prove that there is a constant $C > 0$ such that for all positive integers k and m if P is a set of $n = Ckm$ points in the plane in general position, we can find a convex bundle of size k and width m or a non-crossing family of size m in time $O(n)$. Similarly, we show that we can find a crossing family of size $n/2^{O(\sqrt{\log m})}$ or a non-crossing family of size m in time $O(nm^{1+O((\log m)^{-1/3})})$.

To do so, we use a variant of the semi-algebraic regularity lemma recently proved by Rubin [31]. To state it, we first need to introduce some more definitions.

A Boolean function $\psi\colon \mathbb{R}^{d\times r} \to \{0,1\}$ is an *r-wise semi-algebraic relation* in $\mathbb{R}^d$ if it can be described by a finite combination $(f_1,\dots,f_s,\phi)$, where $f_1,\dots,f_s \in \mathbb{R}[z_1,\dots,z_{rd}]$ are polynomials, and ϕ is a Boolean function in $\{0,1\}^s \to \{0,1\}$, so that

$$\psi(y_1,\dots,y_r) = \phi(f_1(y_1,\dots,y_r) \le 0;\dots;f_s(y_1,\dots,y_r) \le 0)$$

holds for all the ordered r-tuples $(y_1,\dots,y_r)$ of points $y_i \in \mathbb{R}^d$. We call the $(s+1)$-tuple $(f_1,\dots,f_s,\phi)$ a *semi-algebraic description of* ψ (which need not be unique). Such a relation ψ has *description complexity at most* (Δ,s) if it admits a description using at most s polynomials $f_i \in \mathbb{R}[z_1,\dots,z_{rd}]$ of maximum degree at most Δ. We let $\Psi_{d,r,\Delta,s}$ be the family of all such r-wise semi-algebraic relations $\psi\colon \mathbb{R}^{d\times r} \to \{0,1\}$ with description complexity bounded by (Δ,s).

▶ **Theorem 12** ([31]). *The following statement holds for any fixed integers $d \ge 1$, $r \ge 2$, $\Delta \ge 0$, $s \ge 1$, and any fixed $\delta > 0$. Let $X_1\dots,X_r$ be sets of n points in $\mathbb{R}^d$, and ψ be an r-wise relation in $\Psi_{d,r,\Delta,s}$ which is satisfied for at least $\varepsilon|X_1|\cdot\dots\cdot|X_r|$ of the r-tuples $(x_1,\dots,x_r) \in X_1\times\dots\times X_r$. Then one can find, in expected time $O(\sum_{i=1}^r(|X_i|+1/\varepsilon)\log(1/\varepsilon))$, subsets $Y_i \subseteq X_i$, each of cardinality $\Omega(\varepsilon^{d+1+\delta}|X_i|)$, so that $\psi(y_1,\dots,y_r)=1$ holds for all $(y_1,\dots,y_r) \in Y_1\times\dots\times Y_r$.*

The proof of the Same-Type Lemma by Fox, Pach, and Suk [18] is constructive. We go through it step by step in order to estimate the running time of the algorithm it yields. If $(p_1,\dots,p_n)$ is a sequence of $n \ge d+1$ points in $\mathbb{R}^d$ in general position, then the *order-type* of $(p_1,\dots,p_n)$ is the mapping $\chi\colon \binom{P}{d+1} \to \{+1,-1\}$, which assigns to each $(d+1)$-tuple of these points its orientation (positive or negative). For integers $d \ge 2$ and $r \ge d+1$, let $X_1,\dots,X_r$ be pairwise disjoint sets in $\mathbb{R}^d$, each containing n points, whose union is in general position. It is known that the number of different order-types of r-element point sets in $\mathbb{R}^d$ is at most $r^{O(d^2r)}$ [20, 21]. Thus, by the pigeonhole principle, at least $r^{-O(d^2r)}|X_1|\cdots|X_r|$ r-tuples $(x_1,\dots,x_r) \in (X_1,\dots,X_r)$ have the same order-type χ. We define the relation $E \subset X_1\times\dots\times X_r$, where $(x_1,\dots,x_r)\in E$ if and only if $(x_1,\dots,x_r)$ has the order-type χ. Then it can be shown [18] that the description complexity of E is $\left(\binom{r}{d+1},1\right)$. Therefore, we can apply Theorem 12 to the r-partite semi-algebraic hypergraph (P,E) with d, r, $\Delta = \binom{r}{d+1}$, $s=1$, $\delta=1$, and $\varepsilon = r^{-O(d^2r)}$ to obtain subsets $Y_i \subseteq X_i$, each of cardinality $c(d,r)\cdot|X_i|$, with the same-type property, where $c(d,r) \in r^{-O(d^3r)}$, in time

$$O\left(r\left(n + r^{O(d^2r)}\right)\log\left(r^{O(d^2r)}\right)\right) \le O((n+r^{O(d^2r)})d^2r^2\log r).$$

Thus, we obtain the following estimate on the time complexity of the Same-Type Lemma.

▶ **Corollary 13.** *For all integers $d \ge 2$ and $r \ge d+1$, the sets $Y_1,\dots,Y_r$ from Theorem 8 can be found in expected time $O((n+r^{O(d^2r)})d^2r\log r)$ if $|X_1| = \dots = |X_r| = n$.* ◀

▶ **Theorem 6.** *There is a constant $C > 0$ such that for all positive integers k and m, if P is a set of $n = Ckm$ points in the plane in general position, then a convex bundle of size k and width m or a non-crossing family of size m can be computed in expected time $O(n)$.*

Further, a crossing family of size $n/2^{O(\sqrt{\log m})}$ or a non-crossing family of size m can be computed in expected time $O(nm^{1+O((\log m)^{-1/3})})$.

Proof. Let P be a set of at least $\Theta(km)$ points in the plane in general position. In the proof of Theorem 2, we apply the Same-Type Lemma (Theorem 8) with $d = 2$, $r = 6$, $n = \Theta(m)$ $O(k)$-times to obtain a convex bundle $A_1, \ldots, A_k, B_1, \ldots, B_k$ of size $2k$ and width m in P. Since $k = \Theta(|P|/m)$, it follows from Corollary 13 applied with $d = 2$, $r = 6$, and $n = |P|$ that this can be done in time

$$O\left(\frac{|P|}{m} \cdot m\right) = O(|P|),$$

which is the estimate on the time complexity in Theorem 2.

To prove Theorem 4, we also apply Theorem 5 to each set $A_i \cup B_i$ with $1 \leq i \leq k$. The proof of Theorem 5 is an adaptation of the proof of Theorem 1 and has asymptotically the same time complexity. As mentioned by Pach, Rubin, and Tardos [29], their proof yields an algorithm whose running time is $n^{2+O((\log n)^{-1/3})}$ on sets of n points. Since $k = \Theta(|P|/m)$ and $|A_i \cup B_i| \in O(m)$, it follows from this and the previous estimate that the total running time is

$$O(|P| + (|P|/m) \cdot m^{2+O((\log m)^{-1/3})}) = O(|P| \cdot m^{1+O((\log m)^{-1/3})}),$$

which completes the proof. ◀

6 Discussion and open problems

As the main result in this work, we have obtained a Ramsey-type result that gives a lower bound on the size of a maximum crossing family in a set P of n points in general position in the plane in dependence of the size m of a maximum non-crossing family in P. The resulting bound of $n/2^{O(\sqrt{\log m})}$ for the crossing family improves the currently best known lower bound of $n/2^{O(\sqrt{\log n})}$ whenever $m \in o(n)$. Our proofs are constructive and yield an algorithm that finds a crossing family of desired size in polynomial expected time.

The result also constitutes a relevant step towards answering Problem 3. Unfortunately, the obtained bounds are not strong enough to completely resolve Problem 3, which seems difficult. Thus, the first natural open problem is to improve the currently best lower bound $k \in n/2^{O(\sqrt{\log m})}$ on the size k of the largest crossing family determined by a set of n points in the plane in general position that does not contain a non-crossing family of size m. In particular, it is interesting to find out for how large values of m we can get that $k \in \Omega(n)$. It follows from both Corollary 3 and Theorem 4 that if $m \in O(1)$, then $k \in \Omega(n)$. Is it possible to obtain a linear lower bound on k even if $m \in \omega(1)$?

▶ **Problem 4.** *Does there exist a constant $c > 0$ and a function $f : \mathbb{N} \to \mathbb{N}$ with $\lim_{n\to\infty} f(n) = \infty$ such that every set of n points in the plane in general position contains either a crossing family of size at least cn or non-crossing family of size at least $f(n)$?*

Bose, Hurtado, Rivera-Campo, and Wood showed that the edge set of every complete geometric graph with a spoke set of size k can be partitioned into $n - k$ plane trees [12]. Motivated by this result, Schnider [32] posed the following variant of Problem 3.

▶ **Problem 5** ([32])**.** *Is there a constant $c > 0$ such that every set of n points in the plane in general position contains a spoke set of size cn or a non-crossing family of size cn?*

Since every crossing family is a spoke set but not every spoke set is a crossing family, Problem 5 is a weaker variant of Problem 3. However, already an affirmative solution of Problem 5 would imply that the edge set of any complete geometric graph can be partitioned into cn plane subgraphs for some constant $c < 1$.

Finally, the well-known Problem 1, which asks whether or not $T(n) \in \Theta(n)$ [14], still remains open.

References

1 E. Ackerman. On the maximum number of edges in topological graphs with no four pairwise crossing edges. *Discrete Comput. Geom.*, 41(3):365–375, 2009. doi:10.1007/s00454-009-9143-9.

2 E. Ackerman. On topological graphs with at most four crossings per edge. *Comput. Geom.*, 85:101574, 31, 2019. doi:10.1016/j.comgeo.2019.101574.

3 O. Aichholzer, J. Kynčl, M. Scheucher, B. Vogtenhuber, and P. Valtr. On crossing-families in planar point sets. *Comput. Geom.*, 107:Paper No. 101899, 8, 2022. doi:10.1016/j.comgeo.2022.101899.

4 O. Aichholzer, J. Obenaus, J. Orthaber, R. Paul, P. Schnider, R. Steiner, T. Taubner, and B. Vogtenhuber. Edge partitions of complete geometric graphs. In *38th International Symposium on Computational Geometry*, volume 224 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 6, 16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/lipics.socg.2022.6.

5 N. Alon, J. Pach, R. Pinchasi, R. Radoičić, and M. Sharir. Crossing patterns of semi-algebraic sets. *J. Combin. Theory Ser. A*, 111(2):310–326, 2005. doi:10.1016/j.jcta.2004.12.008.

6 P. Angelini, M. A. Bekos, M. Kaufmann, M. Pfister, and Torsten Ueckerdt. Beyond-planarity: Turán-type results for non-planar bipartite graphs. In *29th International Symposium on Algorithms and Computation*, volume 123 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 28, 13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ISAAC.2018.28.

7 P. Angelini, G. Da Lozzo, H. Förster, and T. Schneck. 2-layer k-planar graphs density, crossing lemma, relationships and pathwidth. *Comput. J.*, 67(3):1005–1016, 2024. doi:10.1093/comjnl/bxad038.

8 Patrizio Angelini, Giordano Da Lozzo, Henry Förster, and Thomas Schneck. 2-layer k-planar graphs: Density, crossing lemma, relationships, and pathwidth. In *Graph Drawing and Network Visualization: 28th International Symposium, GD 2020, Vancouver, BC, Canada, September 16–18, 2020, Revised Selected Papers*, pages 403–419, Berlin, Heidelberg, 2020. Springer-Verlag. doi:10.1007/978-3-030-68766-3_32.

9 B. Aronov, P. Erdős, W. Goddard, D. J. Kleitman, M. Klugerman, J. Pach, and L. J. Schulman. Crossing families. *Combinatorica*, 14(2):127–134, 1994. doi:10.1007/BF01215345.

10 I. Bárány and P. Valtr. A positive fraction Erdős–Szekeres theorem. *Discrete Comput. Geom.*, 19(3):335–342, 1998. Dedicated to the memory of Paul Erdős. doi:10.1007/PL00009350.

11 A. Biniaz and A. García. Packing plane spanning trees into a point set. *Comput. Geom.*, 90:101653, 5, 2020. doi:10.1016/j.comgeo.2020.101653.

12 P. Bose, F. Hurtado, E. Rivera-Campo, and D. R. Wood. Partitions of complete geometric graphs into plane trees. *Comput. Geom.*, 34(2):116–125, 2006. doi:10.1016/j.comgeo.2005.08.006.

13 F.-J. Brandenburg. A simple quasi-planar drawing of K_{10}, 2016. URL: https://api.semanticscholar.org/CorpusID:125356339.

14 P. Brass, W. Moser, and J. Pach. *Research problems in discrete geometry*. Springer, New York, 2005.

15 B. Bukh and A. Vasileuski. New bounds for the same-type lemma. *The Electronic Journal of Combinatorics*, 31(2), June 2024. doi:10.37236/12414.

16 V. Capoyleas and J. Pach. A Turán-type theorem on chords of a convex polygon. *J. Combin. Theory Ser. B*, 56(1):9–15, 1992. doi:10.1016/0095-8956(92)90003-G.

17 Adrian Dumitrescu and János Pach. Partitioning Complete Geometric Graphs on Dense Point Sets into Plane Subgraphs. In Stefan Felsner and Karsten Klein, editors, *32nd International*

Symposium on Graph Drawing and Network Visualization (GD 2024), volume 320 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 9:1–9:10, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.GD.2024.9.

18 J. Fox, J. Pach, and A. Suk. A polynomial regularity lemma for semialgebraic hypergraphs and its applications in geometry and property testing. *SIAM J. Comput.*, 45(6):2199–2223, 2016. doi:10.1137/15M1007355.

19 R. Fulek and A. Suk. On disjoint crossing families in geometric graphs. In *Thirty essays on geometric graph theory*, pages 289–302. Springer, New York, 2013. doi:10.1007/978-1-4614-0110-0_15.

20 J. E. Goodman and R. Pollack. The complexity of point configurations. *Discrete Appl. Math.*, 31(2):167–180, 1991. First Canadian Conference on Computational Geometry (Montreal, PQ, 1989). doi:10.1016/0166-218X(91)90068-8.

21 J. E. Goodman and R. Pollack. Allowable sequences and order types in discrete and computational geometry. In *New trends in discrete and computational geometry*, volume 10 of *Algorithms Combin.*, pages 103–134. Springer, Berlin, 1993. doi:10.1007/978-3-642-58043-7_6.

22 M. Hoffmann and C. D. Tóth. Two-planar graphs are quasiplanar. In *42nd International Symposium on Mathematical Foundations of Computer Science*, volume 83 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 47, 14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.MFCS.2017.47.

23 Michael Kaufmann, Boris Klemz, Kristin Knorr, Meghana M. Reddy, Felix Schröder, and Torsten Ueckerdt. The Density Formula: One Lemma to Bound Them All. In Stefan Felsner and Karsten Klein, editors, *32nd International Symposium on Graph Drawing and Network Visualization (GD 2024)*, volume 320 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:17, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.GD.2024.7.

24 D. Lara and C. Rubio-Montiel. On crossing families of complete geometric graphs. *Acta Math. Hungar.*, 157(2):301–311, 2019. doi:10.1007/s10474-018-0880-1.

25 Jiří Matoušek. Efficient partition trees. In *Proceedings of the Seventh Annual Symposium on Computational Geometry*, SCG '91, pages 1–9, New York, NY, USA, 1991. Association for Computing Machinery. doi:10.1145/109648.109649.

26 Jiří Matoušek. Efficient partition trees. *Discrete &; Computational Geometry*, 8(3):315–334, August 1992. doi:10.1007/bf02293051.

27 J. Orthaber. Ramsey on crossing families. Presented at the 19th European Research Week on Geometric Graphs, August 2024. Trier, Germany.

28 J. Orthaber and J. Obenaus. Edge partitions of complete geometric graphs (part 1). https://arxiv.org/abs/2108.05159, 2021.

29 J. Pach, N. Rubin, and G. Tardos. Planar point sets determine many pairwise crossing segments. *Adv. Math.*, 386:Paper No. 107779, 21, 2021. doi:10.1016/j.aim.2021.107779.

30 J. Pach, M. Saghafian, and P. Schnider. Decomposition of geometric graphs into star-forests. In *Graph drawing and network visualization. Part I*, volume 14465 of *Lecture Notes in Comput. Sci.*, pages 339–346. Springer, Cham, 2023. doi:10.1007/978-3-031-49272-3_23.

31 N. Rubin. An efficient regularity lemma for semi-algebraic hypergraphs. https://arxiv.org/abs/2407.15518, 2024.

32 P. Schnider. Edge colorings of complete geometric graphs. Presented at the 19th European Research Week on Geometric Graphs, August 2024. Trier, Germany.

33 P. Valtr. Lines, line-point incidences and crossing families in dense sets. *Combinatorica*, 16(2):269–294, 1996. doi:10.1007/BF01844852.

A Systematic Approach to Crossing Numbers of Cartesian Products with Paths

Zayed Asiri
Flinders University, Adelaide, Australia

Ryan Burdett
Flinders University, Adelaide, Australia

Markus Chimani
Theoretical Computer Science, Osnabrück University, Germany

Michael Haythorpe
Flinders University, Adelaide, Australia

Alex Newcombe
Flinders University, Adelaide, Australia

Mirko H. Wagner
Theoretical Computer Science, Osnabrück University, Germany

Abstract

Determining the crossing numbers of Cartesian products of small graphs with arbitrarily large paths has been an ongoing topic of research since the 1970s. Doing so requires the establishment of coincident upper and lower bounds; the former is usually demonstrated by providing a suitable drawing procedure, while the latter often requires substantial theoretical arguments. Many such papers have been published, which typically focus on just one or two small graphs at a time, and use ad hoc arguments specific to those graphs. We propose a general approach which, when successful, establishes the required lower bound. This approach can be applied to the Cartesian product of any graph with arbitrarily large paths, and in each case involves solving a modified version of the crossing number problem on a finite number (typically only two or three) of small graphs. We demonstrate the potency of this approach by applying it to Cartesian products involving all 133 graphs of orders five or six, and show that it is successful in 128 cases. This includes 60 cases which a recent survey listed as either undetermined, or determined only in journals without adequate peer review.

2012 ACM Subject Classification Mathematics of computing → Graph theory

Keywords and phrases Crossing number, Cartesian graph products, proof framework

Digital Object Identifier 10.4230/LIPIcs.GD.2025.20

Related Version *Previous Version*: `https://arxiv.org/abs/2409.06755`

Supplementary Material *Collection (proof files, validator, and experimental data)*: `https://tcs.uos.de/research/cr` [5]

1 Introduction

In this paper, we will refer to a number of common graph families, and so we list them upfront to aid the reader. P_n, C_n, S_n is the path, cycle, star with n edges and thus on $n+1$, n, $n+1$ vertices respectively. The Cartesian product of two graphs G and H is written as $G \square H$. What results is a graph with vertex set $V(G) \times V(H)$, and edges between vertices (u, u') and (v, v') if and only if either $u = v$ and $(u', v') \in E(H)$, or $u' = v'$ and $(u, v) \in E(G)$.

A *drawing* D of a graph G is a representation of G in the plane, such that each vertex is mapped to a discrete point, and each edge (a, b) is mapped to a closed curve between the points corresponding to vertices a and b, which does not intersect with a point corresponding


© Zayed Asiri, Ryan Burdett, Markus Chimani, Michael Haythorpe, Alex Newcombe, and Mirko H. Wagner;
licensed under Creative Commons License CC-BY 4.0



to any other vertex. If two curves intersect in such a drawing, we say that there is a *crossing* between their corresponding edges, and we denote the number of crossings in the drawing D as $cr_D(G)$. Then the *crossing number problem* (CNP) is to determine $\mathrm{cr}(G) \coloneqq \min_D cr_D(G)$, the minimum number of crossings among all possible drawings of G. We can assume that all intersections are crossings, rather than tangential, and also that three edges never intersect at a common point. Furthermore, a drawing is said to be *simple* if edges incident with a common vertex do not intersect, and no two edges intersect more than once. It is well known that for any graph G, $\mathrm{cr}(G)$ crossings are achieved by a simple drawing.

CNP is known to be NP-hard [18], and is notoriously difficult to solve, even for relatively small graphs. Indeed, $\mathrm{cr}(K_{13})$ and $\mathrm{cr}(K_{9,9})$ are still unknown despite significant effort [1, 15, 33, 42]. Nonetheless, there are some infinite families of graphs for which the crossing numbers are known; these are summarised in the recent survey paper by Clancy et al. [15]. The most common of these are families which result from Cartesian products, the study of which originated with a conjecture by Harary et al. [20] in 1973 that $\mathrm{cr}(C_m \square C_n) = (m-2)n$ for $n \geq m \geq 3$. Despite significant effort [2–4, 6, 17, 19, 32, 35, 36] to date the conjecture has only been resolved for the case when $m \leq 7$, and also the case when $n \geq m(m+1)$. While resolving the $m = 4$ case, Beineke and Ringeisen [6] considered $\mathrm{cr}(G \square C_n)$ for all six non-isomorphic connected graphs G of order four, and successfully determined all of them except for the case $G = S_3$. This latter case was subsequently settled by Jendrol' and Ščerbová [22]. Following this, Klešč [24] determined $\mathrm{cr}(G \square P_n)$ and $\mathrm{cr}(G \square S_n)$ for the same six connected graphs G of order four. This was then followed by a decade-long effort by Klešč [23, 25–29] to determine $\mathrm{cr}(G \square P_n)$ for all 21 non-isomorphic connected graphs G of order five, which was finally completed in 2001. In the two decades since, significant efforts have been made by multiple authors (including Klešč) to extend these results to the 112 non-isomorphic connected graphs of order six. Currently, slightly less than half of these have been resolved; the progress is chronicled in [15].

In order to determine $\mathrm{cr}(G \square P_n)$ for a given graph G, one needs to establish lower and upper bounds, and show that they coincide. Establishing a valid upper bound is generally straight-forward, and is usually either achieved by providing a drawing procedure which results in the desired number of crossings, or else uses an existing result and the monotonicity property $\mathrm{cr}(G \square P_n) \leq cr(H \square P_n)$ if G is a subgraph of H. Establishing a lower bound is typically much more complicated, usually involving ad hoc arguments specific to the graph in question. Due to this ad hoc nature, it is not uncommon for publications to focus on a single graph G and determine $\mathrm{cr}(G \square P_n)$ for that one case (e.g. see [27, 28, 30, 34]). Even in papers which focus on several graphs, often the complicated arguments are needed for only one or two graphs, and the remaining results follow as corollaries.

In this paper, we propose a new approach to determining, $\mathrm{cr}(G \square P_n)$ which can be applied to any graph G. There are two possible outcomes to this approach; either the required lower bound is established, or else nothing is established. We will demonstrate that the approach is successful in resolving 128 of the 133 non-isomorphic connected graphs of orders five or six, including 60 cases which were hitherto unresolved. Of the five remaining graphs, four have been previously solved in the literature, while the one remaining graph is equivalent to K_5 with a pendant edge attached to one of its vertices. We discuss these five graphs further towards the end of Section 5.

The approach we describe requires a modified version of CNP to be solved for $G \square P_d$, where d is a small number (typically $d = 2$ or $d = 3$ suffices). As such, this approach is tractable for sufficiently small graphs G.

The remainder of this paper is laid out as follows. In Section 2 we introduce the modified version of CNP, which we call *binary-weighted capacity-constrained CNP* (BCCNP). In Section 3 we describe the approach which, when successful, uses this modified version of CNP to obtain the required lower bounds for $\mathrm{cr}(G\square P_n)$. In Section 4 we briefly discuss the manner in which we will establish the required upper bounds and base cases, and also the manner in which we solve instances of BCCNP. In Section 5 we report on calculations which test the efficacy of the approach, and show that we are able to determine $\mathrm{cr}(G\square P_n)$ for 60 new graphs G on six vertices, and also explore its efficacy on graphs G with seven vertices. Finally, we conclude the paper in Section 6.

2 Binary-weighted capacity-constrained CNP (BCCNP)

For a given graph G, denote by $C(G) \subseteq E(G) \times E(G)$ the set of all possible pairs of edges between which a crossing may occur in a simple drawing. That is, $C(G)$ contains all pairs of non-adjacent edges. Then, for a simple drawing D of G, and $c \in C(G)$, define a binary variable x_c^D to be equal to 1 if crossing c occurs in D, and 0 otherwise. CNP is then equivalent to

$$\min_D \sum_{c \in C(G)} x_c^D, \text{ over all simple drawings } D.$$

The above formulation considers every possible crossing that could occur in D and then counts those that do. However, suppose that we additionally define some subset $B \subseteq C(G)$, and are only interested in simple drawings that minimize the crossings in B:

$$\min_D \sum_{c \in B} x_c^D, \text{ over all simple drawings } D.$$

This is equivalent to the *weighted CNP* (cf. Schaefer [37]) when all the weights are binary.

Another variant that could be considered would be to add some additional constraints to the formulation. Suppose that we have a family $\mathcal{F} = \{\mathcal{F}_i\}_i$ which contains subsets $\mathcal{F}_i \subseteq C(G)$, and an $|\mathcal{F}|$-dimensional vector $\mathbf{f} = (f_i)$ of non-negative integer capacities. Then a set of constraints of the form $\sum_{c \in \mathcal{F}_i} x_c^D \leq f_i$, for all $1 \leq i \leq |\mathcal{F}|$, could be imposed: we only allow simple drawings that, for each i, contain no more than f_i crossings from the subset $\mathcal{F}_i$. We say $\mathcal{F}_i$ has a *maximum capacity* of f_i.

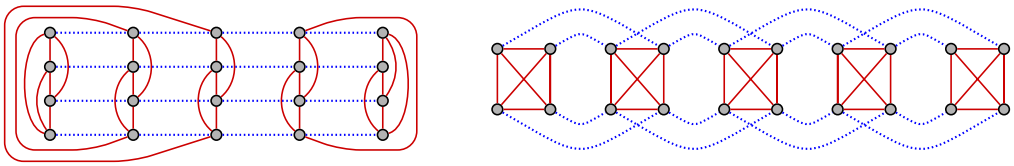
The two concepts above can be combined together; we refer to the resulting problem as *binary-weighted capacity-constrained CNP* (BCCNP), formulated as:

$$\min_D \sum_{c \in B} x_c^D$$

s.t. D is a simple drawing, and

$$\sum_{c \in \mathcal{F}_i} x_c^D \leq f_i \qquad \forall i = 1, \ldots, |\mathcal{F}|$$

An instance of BCCNP requires us to specify not only the graph G, but also the subset B, the family $\mathcal{F} \subseteq 2^X$ of subsets (where $X \subseteq C(G)$), and the capacity vector $\mathbf{f}$. Thus, let $\mathrm{cr}(G, B, \mathcal{F}, \mathbf{f})$ denote the solution of a BCCNP instance. Naturally, $\mathrm{cr}(G, C(G), \emptyset, \emptyset)$ yields the standard CNP. BCCNP has potential applications in its own right. However, our reason for defining it is that it will assist us in determining the crossing numbers for various families of graphs.



(a) Optimal drawing with 8 crossings; the three inner K_4-copies are involved in 2 crossings each.

(b) a-restricted drawing for $a = 2$ with 11 crossings.

Figure 1 Two drawings of $K_4 \square P_4$. The K_4-copies are solid red; the path edges are dotted blue.

3 Using BCCNP to determine crossing numbers

Suppose that we have a graph G, and that we have reason to believe there are integers s, a, b (with $s, a > 0$) such that $\mathrm{cr}(G \square P_n) = an - b$ for $n \geq s$. Certainly this is the case for every established result to date [15]. Furthermore, suppose that we already possess a proof that this formula holds for $n = s, s+1, \ldots, t$ for some integer t, and also that we have established the corresponding upper bound $\mathrm{cr}(G \square P_n) \leq an - b$ for $n \geq s$. Then all that remains is to determine the lower bound, $\mathrm{cr}(G \square P_n) \geq an - b$ for $n > t$.

In order to determine the lower bound, we will seek to look at a fixed-size graph $G \square P_d$ for some small integer d, and use it to infer results about $G \square P_n$ for arbitrarily large n. As such, in what follows, we will tend to use P_d when referring to a path of a fixed size, and P_n otherwise.

We begin by giving some definitions and notation that will be useful in the upcoming discussion. Note that the graph $G \square P_n$ contains $n+1$ copies of G, with consecutive copies linked together by some edges. Denote the copies of G as G^0, G^1, …, G^n, each with edges $E(G^i)$. We call remaining edges *path edges*; those linking G^i to G^{i+1} form the subgraph H^i, for $i = 0, \ldots, n-1$. Note that each H^i corresponds to a matching between G^i and G^{i+1}; see Figure 1a for an example.

▶ **Definition 1.** *Consider a positive integer a. A simple drawing of $G \square P_n$ is* a-restricted *(cf. Figure 1b) if each copy of G has fewer than a crossings on its edges.*

For sets of crossings, we use the shorthand $H' \otimes H'' \coloneqq E(H') \times E(H'')$ and define

$$\mathrm{cr}_a(G, d, B) \coloneqq \mathrm{cr}(G \square P_d, B, \left\{G^i \otimes G \square P_d\right\}_{0 \leq i \leq d}, (a-1) \cdot \mathbf{1}),$$

which is the minimum number of crossings on the set of edge pairs B over all a-restricted drawings of $G \square P_d$.

▶ **Lemma 2.** *Suppose that there is an integer $n \geq 2$ such that $\mathrm{cr}(G \square P_{n-1}) \geq a(n-1) - b$, but $\mathrm{cr}(G \square P_n) < an - b$. Then every crossing-minimal drawing of $G \square P_n$ is a-restricted.*

Proof. Suppose that there is a crossing-minimal drawing D_n of $G \square P_n$ which is not a-restricted. Recall that $G \square P_n$ has $n+1$ copies of G. Since D_n is not a-restricted, at least one of those copies, say G^i, must have a or more crossings on its edges. By deleting the edges $E(G^i)$ from the drawing, we obtain a drawing of a new graph with fewer than $a(n-1) - b$ crossings. However, since this new drawing is of a graph that is either homeomorphic to $G \square P_{n-1}$, or else, if $i = 0$ or $i = n$, contains $G \square P_{n-1}$ as a subgraph, this violates the initial assumption. ◀

Now, consider the graph $G \square P_d$ for some $d \geq 2$. The arguably most natural approach when discussing Cartesian products with paths is to try to consider the graph "copy-wise", i.e., look at the graph in chunks of subgraphs $G_i \cup H_i$, possibly akin to how we define our

	G_0	H_0	G_1	H_1	G_2	H_2	G_3	H_3	G_4	H_4	G_5	H_5	G_6	H_6	G_7
G^0	0	0	1	1	1	1	2	2	2	2	3	3	3	3	4
H^0	0	1	1	1	1	2	2	2	2	3	3	3	3	4	4
G^1	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4
H^1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4
G^2	1	1	2	2	2	2	3	3	3	3	4	4	4	4	5
H^2	1	2	2	2	2	3	3	3	3	4	4	4	4	5	5
G^3	2	2	2	2	3	3	3	3	4	4	4	4	5	5	5
H^3	2	2	2	3	3	3	3	4	4	4	4	5	5	5	5
G^4	2	2	3	3	3	3	4	4	4	4	5	5	5	5	6
H^4	2	3	3	3	3	4	4	4	4	5	5	5	5	6	6
G^5	3	3	3	3	4	4	4	4	5	5	5	5	6	6	6
H^5	3	3	3	4	4	4	4	5	5	5	5	6	6	6	6
G^6	3	3	4	4	4	4	5	5	5	5	6	6	6	6	7
H^6	3	4	4	4	4	5	5	5	5	6	6	6	6	7	7
G^7	4	4	4	4	5	5	5	5	6	6	6	6	7	7	7

Figure 2 Crossing bands A_i^7 for $G\square P_7$ and $i = 0, \ldots, 7$, visually highlighted as colored stripes. A colored cell represents crossings between an edge of the subgraph corresponding to the row and the column, respectively. The number "i" in a cell indicates that these crossings are contained in A_i^7. In contrast to this, the restriction of an a-restricted drawing w.r.t. copy G^i sums over all crossings in row and column G^i; a bold outline marks this for G^3. This is very different from A_3^7.

a-restrictions. However, it turns out that a different way to group crossings is more useful in our proofs. We partition the possible crossings $C(G\square P_d)$ into $d+1$ *crossing bands* A_i^d, as follows.

$$\begin{array}{lll} \text{Crossings in } G^i \otimes (G^j \cup H^j) & \text{are assigned to} & A_k^d, \text{ where } k = \lceil (i+j)/2 \rceil, \\ \text{Crossings in } H^i \otimes H^j & \text{are assigned to} & A_k^d, \text{ where } k = \lceil (i+j+1)/2 \rceil. \end{array}$$

Of course, there are other ways the crossings could have been partitioned, and indeed, we experimented with many other choices. The partitioning method we present here is the one we found to be the most effective in obtaining results. The partition of crossings into the crossing bands is visualized in Figure 2 for the case when $d = 7$. For the sake of convenience, in the following discussions we will often refer to a drawing as having crossing bands; technically, this refers to the crossing bands of the underlying graph.

We will show that each crossing band contributes a specific minimum number of crossings to the overall crossing number of $G\square P_d$. Intuitively speaking, one may think about a natural drawing, like Figure 1a: Essentially each G-copy is drawn identically and requires a certain number of crossings, except for the left- and right-most G-copy ("front" and "end") which may require less crossings. However, for more complex G, the special front and end may consist of multiple G-copies each. All path-products for which the crossing number is known allow such a "natural" optimal drawing, however, our proof may of course not assume that this would always be the case. Nonetheless, this pattern allows us to establish certain minimal numbers of crossings that are *forced* along the crossing bands.

Let $d \geq 2$ and $0 < m < d$. We use the term *middle force* to refer to the minimum possible number of crossings from A_m^d in any a-restricted drawing of $G\square P_d$. Likewise, the term *front force* (*end force*) refers to the minimum possible number of crossings from the first m (final

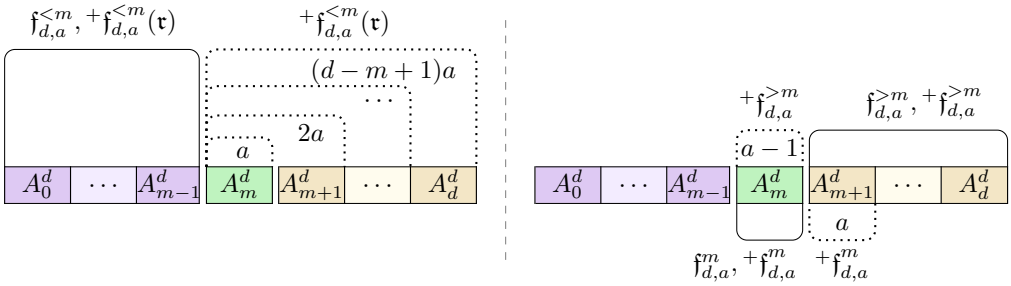


Figure 3 Crossing bands considered in a (plus-)force triple: **(left)** front, **(right)** middle and end.

$d-m$, respectively) crossing bands in any a-restricted drawing of $G\square P_d$ (cf. Figure 3):

$$\begin{aligned}
&\textit{front force} && \mathfrak{f}^{<m}_{d,a} &&:= \quad \mathrm{cr}_a(G,d,\quad A^d_0 \cup \cdots \cup A^d_{m-1} \quad),\\
&\textit{middle force} && \mathfrak{f}^{m}_{d,a} &&:= \quad \mathrm{cr}_a(G,d,\quad\quad A^d_m \quad\quad),\\
&\textit{end force} && \mathfrak{f}^{>m}_{d,a} &&:= \quad \mathrm{cr}_a(G,d,\quad A^d_{m+1} \cup \cdots \cup A^d_d \quad).
\end{aligned}$$

Together they define the *force triple* $(\mathfrak{f}^{<m}_{d,a}, \mathfrak{f}^{m}_{d,a}, \mathfrak{f}^{>m}_{d,a})$. The rationale for considering the force triple is revealed in the following lemma.

▶ **Lemma 3.** *Let $(\mathfrak{f}^{<m}_{d,a}, \mathfrak{f}^{m}_{d,a}, \mathfrak{f}^{>m}_{d,a})$ be a force triple of G. Let D_n, with $n \geq d$, be an a-restricted drawing of $G\square P_n$. Then there are*

(a) *at least $\mathfrak{f}^{<m}_{d,a}$ crossings in D_n from the first m crossings bands;*
(b) *at least $\mathfrak{f}^{m}_{d,a}$ crossings in D_n from the $(m+i)$-th crossing band for $i \in \{0,\dots,n-d\}$; and*
(c) *at least $\mathfrak{f}^{>m}_{d,a}$ crossings in D_n from the final $d-m$ crossing bands.*

Proof. First, note that D_n contains $n-d+1$ subdrawings of $G\square P_d$. In particular, we denote the j-th subdrawing as D^j_n, for $j \in \{0,\dots,n-d\}$; it is the one containing copies G^j through to G^{j+d}. Since D_n is an a-restricted drawing, each of these subdrawings is also a-restricted. We can obtain lower bounds on how many crossings in D_n come from each crossing band, by looking at the forced number of crossings in corresponding subdrawings.

Consider D^0_n, and recall that it is a-restricted. Then, by definition, the number of crossings in D^0_n from its first m crossing bands is at least $\mathfrak{f}^{<m}_{d,a}$. However, the set of potential crossings contained in its first m crossing bands is a strict subset of the potential crossings contained in the first m crossing bands of D_n. Hence, item (a) is true. Analogous arguments hold for item (b) by considering the m-th crossing bands in D^i_n for $i \in \{0,\dots,n-d\}$, and for item (c) by considering the final $d-m$ crossing bands in D^{n-d}_n. ◀

Lemmas 2 and 3 lead to our first main statement.

▶ **Theorem 4.** *Let $(\mathfrak{f}^{<m}_{d,a}, \mathfrak{f}^{m}_{d,a}, \mathfrak{f}^{>m}_{d,a})$ be a force triple of G. If*

$$\mathrm{cr}(G\square P_{d-1}) \geq a(d-1)-b, \qquad \mathfrak{f}^{<m}_{d,a} + \mathfrak{f}^{>m}_{d,a} \geq a(d-1)-b, \qquad \textit{and} \qquad \mathfrak{f}^{m}_{d,a} \geq a,$$

then for all $n \geq d$ we also have

$$\mathrm{cr}(G\square P_n) \geq an - b.$$

Proof. We will prove by induction, where the base case is known to be true according to the assumptions. Then, suppose that there is some value $n \geq d$ such that $\mathrm{cr}(G\square P_{n-1}) \geq a(n-1)-b$, but $\mathrm{cr}(G\square P_n) < an-b$. Let D_n be a crossing-minimal drawing of $G\square P_n$. By Lemma 2, D_n is a-restricted, and we note that it has fewer than $an-b$ crossings. Then, by Lemma 3, D_n must contain at least $\mathfrak{f}^{<m}_{d,a} + \mathfrak{f}^{>m}_{d,a} + (n-d+1)\mathfrak{f}^{m}_{d,a} \geq an-b$ crossings, contradicting our initial assumption and completing the proof. ◀

As will be demonstrated in Section 5, Theorem 4 is already sufficient to provide the desired lower bounds in several cases. However, in many cases the force triple values are not large enough to meet the conditions of Theorem 4. In Section 3.1 we briefly discuss why this might occur, and then propose a more sophisticated way of considering the forces which leads to significantly improved results.

3.1 Plus-forces and star-forces

The above results are obtained by partitioning all possible crossings of $G\square P_n$ into various crossing bands, and then considering lower bounds on the number of crossings from those crossing bands. If each of these individual lower bounds is sufficiently large, we obtain the desired result. However, in practice, it is often possible to draw a graph such that there are relatively few crossings in one crossing band, at the cost of a higher number of crossings in nearby crossing bands. Theorem 4 does not take this into account and pessimistically assumes that the minima for each crossing band could be attained simultaneously. Whenever this is not the case, the conditions of Theorem 4 are not met and it cannot be applied.

To remedy this, we propose a more sophisticated way of measuring forces. When calculating the above force values by solving a BCCNP instance, the objective function only considers crossings from certain crossing band(s); except for insisting on an a-restricted drawing, there are no further restrictions on the other crossing bands. This leads to the issue mentioned above, where additional crossings are able to "hide" in the other, largely unrestricted, crossing bands. With this in mind, we introduce *plus-forces*, which are illustrated in Figure 3. Each plus-force is defined the same as its corresponding force, but with additional restrictions that are placed on the number of crossings in other crossing bands. In particular, for the middle and end plus-forces, restrictions are placed on exactly one, adjacent, crossing band. For the front plus-force, restrictions are placed on $\mathfrak{r}$ crossing bands, for $\mathfrak{r} \in \{0, \ldots, d-m+1\}$. Hence, we use the terms ${}^{+}\mathfrak{f}^{<m}_{d,a}(\mathfrak{r})$, ${}^{+}\mathfrak{f}^{m}_{d,a}$, ${}^{+}\mathfrak{f}^{>m}_{d,a}$ to denote the front, middle, and end plus-forces, respectively. We again require $d \geq 2$ and $0 < m < d$:

The *middle plus-force* ${}^{+}\mathfrak{f}^{m}_{d,a}$ is the minimum number of crossings from A^d_m in any a-restricted drawing, such that there are at most a crossings from A^d_{m+1}. This can be computed analogously to $\mathfrak{f}^{m}_{d,a}$, by simply adding the one extra capacity constraint $\sum_{c \in A^d_{m+1}} x^D_c \leq a$.

The *end plus-force* ${}^{+}\mathfrak{f}^{>m}_{d,a}$ is the minimum number of crossings from the final $d-m$ crossing bands in any a-restricted drawing, such that there are at most $a-1$ crossings from A^d_m. This can be computed in the same way as $\mathfrak{f}^{>m}_{d,a}$ by adding $\sum_{c \in A^d_m} x^D_c \leq a-1$. We specifically clarify that the right-hand side here is really $a-1$, not a as in the middle plus-force, or a multiple of a as in the front plus-force below.

For the *front plus-force* ${}^{+}\mathfrak{f}^{<m}_{d,a}(\mathfrak{r})$, we need to specify $\mathfrak{r} \in \{0, \ldots, d-m+1\}$, which is the number of crossing bands (starting from A^d_m) that restrictions should be placed on. Hence, we define ${}^{+}\mathfrak{f}^{<m}_{d,a}(0) := \mathfrak{f}^{<m}_{d,a}$, and for $\mathfrak{r} > 0$, ${}^{+}\mathfrak{f}^{<m}_{d,a}(\mathfrak{r})$ is the minimum number of crossings from the first m crossing bands in any a-restricted drawing, such that, for every $r \in \{0, 1, \ldots, \mathfrak{r}-1\}$,

there are at most $(r+1)\cdot a$ crossings in $\bigcup_{i=m}^{m+r} A_i^d$. Hence, ${}^{+}\mathfrak{f}_{d,a}^{<m}(\mathfrak{r})$ can be computed in the same way we compute $\mathfrak{f}_{d,a}^{<m}$ by adding $\mathfrak{r}$ additional capacity constraints:

$$\sum_{i=m}^{m+r}\sum_{c\in A_i^d} x_c^D \le (r+1)\cdot a, \qquad \forall r\in\{0,1,\dots,\mathfrak{r}-1\}.$$

Since the plus-forces are simply more restricted versions of the forces, each plus-force is at least as large as its corresponding force. Whether or not they coincide tells us something about a-restricted drawings of the underlying graph. For example, if $\mathfrak{f}_{d,a}^{m} < {}^{+}\mathfrak{f}_{d,a}^{m}$, it tells us that in order for an a-restricted drawing of $G\square P_d$ to have only $\mathfrak{f}_{d,a}^{m}$ crossings from A_m^d, it must have more than a crossings from A_{m+1}^d.

To see why such an observation could be valuable, recall that for Theorem 4, we require $\mathfrak{f}_{d,a}^{m} \ge a$. This requirement ensures there are at least a crossings in each of the "middle" crossing bands (those after the first m and before the final $d-m$ crossing bands), hence ensuring that, for sufficiently large n, drawings of $G\square P_{n+1}$ must have at least a more crossings than drawings of $G\square P_n$. However, the situation could arise where one (or more) of the middle crossing bands has $a-1$ crossings, and yet we would still end up with the expected number of crossings as long as another middle crossing band has at least $a+1$ crossings. Indeed, if $\mathfrak{f}_{d,a}^{m} = a-1$ and ${}^{+}\mathfrak{f}_{d,a}^{m} \ge a$, it tells us that whenever there are $a-1$ crossings in a middle crossing band, the subsequent crossing band (if one exists) must have at least $a+1$ crossings. This observation permits us to adopt a broader condition; we want either (a) $\mathfrak{f}_{d,a}^{m} \ge a$, or (b) $\mathfrak{f}_{d,a}^{m} = a-1$ and ${}^{+}\mathfrak{f}_{d,a}^{m} \ge a$.

However, there are some awkward edge-cases to handle here. For example, what if it is the last middle crossing band that has $a-1$ crossings? Since there is no subsequent middle crossing band (the next crossing band is part of the final $d-m$ crossing bands), we cannot ensure that it has $a+1$ crossings simply by referring to the middle plus-force. Such edge-cases need to be handled delicately. In order to do so, we define the following *star-forces*, in a manner that will make the upcoming proofs neater.

front star-force	${}^{*}\mathfrak{f}_{d,a}^{<m}(\mathfrak{r})$	$:=\min\{$	${}^{+}\mathfrak{f}_{d,a}^{<m}(\mathfrak{r})$		,	$\mathfrak{f}_{d,a}^{<m}$	$+1$ $\}$
middle star-force	${}^{*}\mathfrak{f}_{d,a}^{m}$	$:=\min\{$	${}^{+}\mathfrak{f}_{d,a}^{m}$		,	$\mathfrak{f}_{d,a}^{m}$	$+1$ $\}$
end star-force	${}^{*}\mathfrak{f}_{d,a}^{>m}$	$:=\min\{$	${}^{+}\mathfrak{f}_{d,a}^{>m}$	-1 ,		$\mathfrak{f}_{d,a}^{>m}$	$\}$

In the upcoming Theorem 6, the star-forces, as defined above, will allow us to handle the situation where $\mathfrak{f}_{d,a}^{m} = a-1$ and ${}^{+}\mathfrak{f}_{d,a}^{m} \ge a$, while also taking care of the edge-cases. First, however, we establish a set of intermediate results in Lemma 5. In particular, we give bounds on the number of crossings that must occur in the first m and final $d-m$ crossing bands, and also discuss the manner in which crossings may occur in the middle crossing bands.

▶ **Lemma 5.** *Let* $({}^{*}\mathfrak{f}_{d,a}^{<m}(\mathfrak{r}), {}^{*}\mathfrak{f}_{d,a}^{m}, {}^{*}\mathfrak{f}_{d,a}^{>m})$ *be a star-force triple of* G*, and suppose that* ${}^{*}\mathfrak{f}_{d,a}^{m} \ge a$*. Let* D_n *be an* a*-restricted drawing of* $G\square P_n$ *for* $n \ge d$*. Then:*

- **(a)** *D_n contains at least ${}^{*}\mathfrak{f}_{d,a}^{<m}(\mathfrak{r}) - 1$ crossings from the first m crossing bands;*
- **(b)** *D_n contains at least $a-1$ crossings from A_{i+m}^n for $i\in\{0,\dots,n-d\}$;*
- **(c)** *if D_n contains exactly $a-1$ crossings from A_{i+m}^n for $i\in\{0,\dots,n-d-1\}$, then D_n contains at least $a+1$ crossings from A_{i+m+1}^n; and*
- **(d)** *Let $z=1$ if D_n contains exactly $a-1$ crossings from A_{n-d+m}^n, and $z=0$ otherwise. Then, D_n contains at least ${}^{*}\mathfrak{f}_{d,a}^{>m} + z$ crossings from the final $d-m$ crossing bands.*

Proof.

Item (a). As previously noted, D_n contains $n-d+1$ subdrawings of $G\square P_d$, which we denote as D_n^j for $j \in \{0,\dots,n-d\}$. From Lemma 3(a), we have that D_n contains at least $\mathfrak{f}_{d,a}^{<m}$ crossings from the first m crossing bands. From the definition of ${}^*\mathfrak{f}_{d,a}^{<m}(\mathfrak{r})$, it is clear that $\mathfrak{f}_{d,a}^{<m} \geq {}^*\mathfrak{f}_{d,a}^{<m}(\mathfrak{r}) - 1$. Hence, item (a) is true.

Item (b). By Lemma 3(b), we have that D_n contains at least $\mathfrak{f}_{d,a}^m$ crossings from A_{i+m}^n for $i \in \{0,\dots,n-d\}$. Now consider ${}^*\mathfrak{f}_{d,a}^m$, which by definition is either equal to ${}^+\mathfrak{f}_{d,a}^m$ or $\mathfrak{f}_{d,a}^m + 1$. Since ${}^+\mathfrak{f}_{d,a}^m \geq \mathfrak{f}_{d,a}^m$ and both are integers, there are only two possibilities; either $\mathfrak{f}_{d,a}^m = {}^*\mathfrak{f}_{d,a}^m$, or $\mathfrak{f}_{d,a}^m = {}^*\mathfrak{f}_{d,a}^m - 1$. Either way, item (b) is true.

Item (c). Now, suppose that D_n contains exactly $a-1$ crossings from A_{i+m}^n, which is only possible if $\mathfrak{f}_{d,a}^m = {}^*\mathfrak{f}_{d,a}^m - 1 \geq a-1$. This, in turn, is only possible if ${}^+\mathfrak{f}_{d,a}^m > \mathfrak{f}_{d,a}^m$, and so ${}^+\mathfrak{f}_{d,a}^m \geq a$. According to the definition of ${}^+\mathfrak{f}_{d,a}^m$, it is hence impossible to draw $G\square P_d$ with fewer than a crossings from A_m^d and fewer than $a+1$ crossings from A_{m+1}^d. However, in the subdrawing D_n^i there are fewer than a crossings from A_m^d. Hence, there must be at least $a+1$ crossings in D_n^i from A_{m+1}^d, which implies that there are at least $a+1$ crossings in D_n from A_{i+m+1}^n. Therefore item (c) is true.

Item (d). Finally, by Lemma 3(c), we have that D_n contains at least $\mathfrak{f}_{d,a}^{>m}$ crossings from the final $d-m$ crossing bands. Note that by definition we have $\mathfrak{f}_{d,a}^{>m} \geq {}^*\mathfrak{f}_{d,a}^{>m}$, so if $z=0$ then item (d) is true. Suppose that $z=1$, and item (d) is false. This could only be the case if $\mathfrak{f}_{d,a}^{>m} = {}^*\mathfrak{f}_{d,a}^{>m}$, which implies that D_n must have exactly ${}^*\mathfrak{f}_{d,a}^{>m}$ crossings from the final $d-m$ crossing bands, i.e., one crossing less than suggested by item (d). However, since $z=1$, D_n contains exactly $a-1$ crossings from A_{n-d+m}^n by definition. Thus, the subdrawing D_n^{n-d} must also have no more than $a-1$ crossings in its m-th crossing band, and since D_n^{n-d} is an a-restricted drawing, it must have at least ${}^+\mathfrak{f}_{d,a}^{>m}$ crossings from its final $d-m$ crossing bands. This implies that D_n must contain at least ${}^+\mathfrak{f}_{d,a}^{>m}$ crossings from its final $d-m$ crossing bands. However, by definition we have ${}^+\mathfrak{f}_{d,a}^{>m} \geq {}^*\mathfrak{f}_{d,a}^{>m} + 1 = {}^*\mathfrak{f}_{d,a}^{>m} + z$. Hence item (d) is true. ◀

Lemma 5 enables us to establish a stronger version of Theorem 4.

▶ **Theorem 6.** *Let* $({}^*\mathfrak{f}_{d,a}^{<m}(\mathfrak{r}), {}^*\mathfrak{f}_{d,a}^m, {}^*\mathfrak{f}_{d,a}^{>m})$ *be a star-force triple of* G *and* $\mathfrak{r}' \coloneqq \max(\mathfrak{r},1)$*. If*

$$\mathrm{cr}(G\square P_{d+\mathfrak{r}'-2}) \geq a(d+\mathfrak{r}'-2)-b, \qquad {}^*\mathfrak{f}_{d,a}^{<m}(\mathfrak{r}) + {}^*\mathfrak{f}_{d,a}^{>m} \geq a(d-1)-b, \qquad \text{and} \quad {}^*\mathfrak{f}_{d,a}^m \geq a$$

then for $n \geq d+\mathfrak{r}'-1$*,*

$$\mathrm{cr}(G\square P_n) \geq an-b.$$

Proof. Suppose that there is some value $n \geq d+\mathfrak{r}'-1$ such that $\mathrm{cr}(G\square P_{n-1}) \geq a(n-1)-b$, but $\mathrm{cr}(G\square P_n) < an-b$. Let n be minimal with this property. Let D_n be a crossing-minimal drawing of $G\square P_n$, then D_n contains fewer than $an-b$ crossings, and from Lemma 2 it is an a-restricted drawing. We want to count the number of crossings in D_n, cf. Figure 4. We will use the term *front crossings* to refer to the crossings in D_n from the first m crossing bands, *end crossings* to refer to the crossings in D_n from the final $d-m$ crossing bands, and *middle crossings* to refer to all remaining crossings of D_n. To assist in calculating these, let $\mathfrak{c}_i^j$ be the number of crossings in D_n from $A_{m+i}^n \cup \ldots \cup A_{m+j}^n$. To simplify the notation, let $\mathfrak{c}_i \coloneqq \mathfrak{c}_i^{n-d}$. Note that the number of middle crossings in D_n is equal to $\mathfrak{c}_0$. We seek to compute a lower bound for $\mathfrak{c}_i$ for $i \in \{0,\dots,n-d\}$. As in Lemma 5 it will be convenient to set $z=1$ if D_n has exactly $a-1$ crossings from A_{n-d+m}^n, and $z=0$ otherwise.

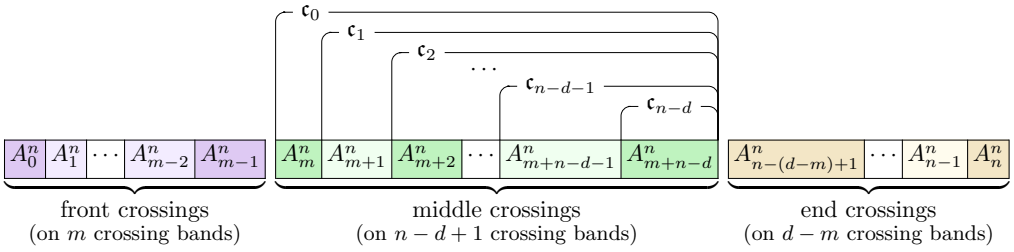


Figure 4 Counting crossings in D_n.

Recall from Lemma 5(b) that $\mathfrak{c}_i^i \geq a-1$, for $i \in \{0,\dots,n-d\}$. If equality holds for any $i < n-d$, then from Lemma 5(c) we know that $\mathfrak{c}_i^{i+1} \geq (a-1)+(a+1) = 2a$. Inductively then, it is clear that $\mathfrak{c}_i^j \geq (j-i+1)a-1$ for $i,j \in \{0,\dots,n-d\}$ and $i \leq j$, and if equality is met it implies that D_n contains exactly $a-1$ crossings from A_{j+m}^n. Hence, by setting $j = n-d$, we obtain

$$\mathfrak{c}_i \geq (n+1-d-i)a - z, \qquad \forall i \in \{0,\dots,n-d\}. \tag{1}$$

In particular, setting $i=0$ in (1) tells us that there are at least $\mathfrak{c}_0 = (n+1-d)a-z$ middle crossings, while Lemma 5(a) and Lemma 5(d) give us lower bounds on the front and end crossings respectively. Combining all of these, we can see that D_n contains at least $({}^*\mathfrak{f}_{d,a}^{<m}(\mathfrak{r})-1) + \mathfrak{c}_0 + ({}^*\mathfrak{f}_{d,a}^{>m} + z) \geq an-b-1$ crossings. Since by assumption D_n has fewer than $an-b$ crossings, it must have exactly $an-b-1$ crossings. In particular, D_n has exactly ${}^*\mathfrak{f}_{d,a}^{<m}(\mathfrak{r})-1$ front crossings. However, from the definition of ${}^*\mathfrak{f}_{d,a}^{<m}(\mathfrak{r})$, this is only possible if ${}^+\mathfrak{f}_{d,a}^{<m}(\mathfrak{r}) > \mathfrak{f}_{d,a}^{<m}$ and $\mathfrak{r} \geq 1$. Hence, D_n must have fewer than ${}^+\mathfrak{f}_{d,a}^{<m}(\mathfrak{r})$ front crossings, and so from the definition of ${}^+\mathfrak{f}_{d,a}^{<m}(\mathfrak{r})$, there must be some positive integer $j \leq \mathfrak{r}$ such that $\mathfrak{c}_0^{j-1} \geq aj+1$. Note that since $n \geq d+\mathfrak{r}'-1$, and $\mathfrak{r} \leq \mathfrak{r}'$, we have $j \leq n-d+1$. If $j = n-d+1$ then $\mathfrak{c}_0 = \mathfrak{c}_0^{j-1} \geq a(n-d+1)+1$. Alternatively, if $j < n-d+1$, then from (1) we have $\mathfrak{c}_j \geq (n+1-d-j)a-z$, and so $\mathfrak{c}_0 = \mathfrak{c}_0^{j-1} + \mathfrak{c}_j \geq (n-d+1)a-z+1$.Either way, adding $\mathfrak{c}_0$ to $({}^*\mathfrak{f}_{d,a}^{<m}(\mathfrak{r})-1) + ({}^*\mathfrak{f}_{d,a}^{>m}+z)$ implies that D_n contains at least $an-b$ crossings, contradicting our initial assumption and completing the proof. ◀

As will be shown in Section 5, Theorem 6 is much stronger than Theorem 4, and will enable us to established the desired lower bounds for many more graph classes.

4 Upper bounds, base cases, and computing forces by solving BCCNP

Upper bounds. An important ingredient of the crossing number proofs is establishing an upper bound matching the lower bound. In practice, in the case of $G\square P_n$, this turns out to be surprisingly simple, as simple enumeration schemes for simple drawings suffice. In fact, in most cases all but the very first and the very last copy of G can be drawn identically, and these drawings are easy to figure out. Thus, we will not discuss this further in the following, but state that whenever we could prove a lower bound, we also identified a construction for the matching upper bound. The tables in the appendix show the corresponding subdrawings (and explain their interpretation).

Base cases and force computation. To apply our above theorems, each graph class $G\square P_n$ requires us to establish (a) $\mathrm{cr}(G\square P_n)$ for small values of n as the base cases, and (b) G-specific force values: sound proofs of BCCNP for small values d, m. Each thereby considered graph is relatively small (typically below 50 edges), but since our goal is to solve many different graph classes, there are too many such instances to reasonably do them all by hand.

It was shown in Buchheim et al. [10] that CNP can be formulated as an integer linear program (ILP). Subsequently, Chimani et al. [11, 13] implemented this formulation as a practical algorithm, which requires sophisticated column generation techniques and other speed-up heuristics, to be tractable for "real-world" graphs with up to 100 edges. As this implementation is generally not easy to understand and validate, Chimani and Wiedera [14] presented a proof and validation framework that extracts a crossing number proof from the ILP computation; this certificate can be independently validated (with comparably simple methods). This tool has since regularly been used to establish base cases or to validate results, e.g., [8, 9, 15, 16, 21, 31, 38–41]. The modifications necessary to tackle BCCNP instead of CNP are relatively straight-forward: analogously to the case of the *simultaneous crossing number* [12], when restricting the objective function to disregard certain crossings, these crossing variables are considered with a small enough coefficient ε instead of 0, to avoid issues with non-simple drawings. The capacity constraints can directly be added to the ILP computation and the proof validator as well. We call this algorithm the *BCCNP solver*; all corresponding certificates for the crossing number proofs required below (as well as the proof validator), can be found at [5].

5 Calculations

Theorems 4 and 6 are only useful when the conditions are met, and checking if they are met requires us to compute multiple forces using the BCCNP solver for some setting of the parameters. Clearly, there are three possible outcomes to this approach. The first outcome is that the values computed by the BCCNP solver satisfy the requirements of at least one of the two theorems, and the desired result is obtained. The second outcome is that the values computed by the BCCNP solver do not satisfy the requirements of either theorem, and no result is obtained. The third outcome is that the BCCNP solver is unable to compute the values, as the graph is too large or complex to be tackled with reasonable computing power[1].

To test the efficacy of the two proposed approaches, we ran them both on all 21 non-isomorphic connected graphs on five vertices, and all 112 non-isomorphic connected graphs on six vertices. Since Theorems 4 and 6 attempt to show that $\mathrm{cr}(G\square P_n) \geq an - b$, we began by assuming that $\mathrm{cr}(G\square P_n) = an - b$ for some integers a, b. We computed $\mathrm{cr}(G\square P_n)$ for some small values of n, and used these to predict the values of a, b. Then, we checked to see if they matched the upper bound yielded from our simple enumeration schemes. Indeed, in all tested cases, the values of a, b matched.

Then, armed with an appropriate choice of a, b, we computed the forces for the (d, m) pair $(2, 1)$, and checked to see if they satisfy the relevant conditions for Theorem 4. If not, we then proceeded to compute the plus-forces for the same (d, m) pair and $\mathfrak{r} = d - m + 1$. Then, we checked if the relevant conditions of Theorem 6 were satisfied first for $\mathfrak{r} = 0$ (by using ${}^{+}\mathfrak{f}_{d,a}^{<m}(0) = \mathfrak{f}_{d,a}^{<m}$), and if not, then for $\mathfrak{r} = d - m + 1$. If any of these checks were successful, we computed the relevant base cases to settle the instance; note that checking for $\mathfrak{r} = 0$ first is

[1] All computations were conducted on an Intel Xeon Gold 6134 with a time limit of 2 hours. Nearly 3/4 of the calculations finished in under a minute and less than 0.05% needed more than half an hour but less than two hours. This indicates that neither more computation power nor a higher time limit is likely to yield significant improvements.

Table 1 The number of instances (as well as percentage) for which Theorems 4 and 6 yield positive results for at least one (d, m) pair $(2, 1)$, $(3, 1)$, or $(4, 2)$.

$G \square P_n$, where G are	Theorem 4		Theorem 6 with $\mathfrak{r} = 0$		Theorem 6 with $\mathfrak{r} = d - m + 1$		total	
5-vertex graphs	19	90%	21	100%	21	100%	**21**	**100%**
6-vertex graphs	91	81%	104	93%	105	94%	**107**	**96%**
7-vertex graphs	529	62%	573	67%	675	79%	**676**	**79%**

worthwhile because the corresponding base case is considerably smaller than for $\mathfrak{r} = n - d + 1$. If none of the checks were successful, then we then repeated the above for the (d, m) pairs $(3, 1)$, and $(4, 2)$, only declaring failure if all of these proved unsuccessful.

We summarize the results of these calculations in Table 1. In particular, we note that we obtained a successful result for all 21 graphs on five vertices, and 107 (out of 112) graphs on six vertices. The successful instances include 60 graphs on six vertices which, according to the recent survey by Clancy et al. [15], have not previously been handled in the literature[2].

Concisely, we establish the following results w.r.t. 6-vertex graphs:

▶ **Theorem 7.** *Let i, in G_i^6, be the graph indices from [15] to name all 6-vertex graphs[3]. For large enough n (in most cases $n \geq 1$), the crossing number $\mathrm{cr}(G_i^6 \square P_n)$ is given by:*

$\mathrm{cr}(G_i^6 \square P_n)$	*i (new results)*	*i (previously established, reproved)*
$0n + 0$		$25, 40$
$1n - 1$		$26, 28, 41, 43, 46, 60$
$2n - 4$		42
$2n - 2$	$49,$	$27, 29, 44, 45, 47, 53, 54, 59, 63, 64, 66, 74, 77, 83, 94$
$2n + 0$		$61, 75, 85$
$3n - 5$		62
$3n - 3$	$67, 76, 78, 92,$	$51, 65, 70, 89, 90, 120$
$3n - 1$	$84, 95, 98, 110$	$68, 71, 86, 87, 91, 111$
$4n - 4$		$31, 72, 79, 80, 104, 113$
$4n - 2$	$82, 88, 99, 100, 101, 108, 115, 116, 118, 133$	
$4n$	$93, 109, 112, 121, 130, 137$	
$5n - 3$	$81, 102, 106, 107, 114, 122, 123, 124, 127,$	125
$5n - 1$	$131, 138, 146$	
$6n - 4$	134	
$6n - 2$	$105, 126, 128, 129, 132, 140, 141, 143,$	103
$6n$	152	
$7n - 1$	$119, 135, 139, 145, 148$	
$8n - 2$	$144, 147, 151$	
$9n - 3$	150	
$9n - 1$	154	
$10n$	$149, 153$	
$12n$	155	

The table includes all 112 connected 6-vertex graphs except for $i = 31, 48, 73, 142, 156$.

[2] This includes instances which have not been handled at all in the literature, and also instances which have only been handled in journals or periodicals which Clancy et al. [15] determined "impose no peer review, or that which does occur is inadequate." They note that such results "should be revisited and submitted to thorough peer review". As such, we treat such results as new in this manuscript.

[3] The index scheme from [15] includes the 44 disconnected 6-vertex graphs, and hence goes up to 156.

The appendix lists all these graphs, together with their upper bound construction and the required values for $d, m, \mathfrak{r}$. As mentioned above, the computer generated proof files for the BCCNP base cases (together with a proof validator) can be found at [5]. It is worth noting that in 78 of the 112 cases the smallest setting, $d = 2$ and $m = 1$, already suffices to prove the claim with Theorem 6 and in 66 cases, even with Theorem 4.

We note that, of the five graphs on six vertices we were unable to obtain a solution, four have already been handled in the literature. In particular, the graph with index $i = 31$ was handled in Bokal [7], the graphs with indices $i = 48, 73$ were handled in Klešč and Petrillová [30], and the graph with index $i = 156$ was handled in Zheng et al. [43]. As such, there is only one graph remaining from the six-vertex set that is yet to be handled in the literature at all, specifically the graph with index $i = 142$. It is equivalent to K_5 with a pendant edge attached to one of the vertices, and based on our experiments it appears the crossing number should be $9n - 3$. We discuss some possible reasons why our approach may have proved unsuccessful for these five graphs in Section 6.

Finally, we also tackled the Cartesian products of paths with any connected 7-vertex graph. There are few existing crossing number results on these graph classes (nine, according to [15]). Our goal here is to determine how many of these graph classes are in fact relatively simple – in terms of solvability using our framework – and thus do not merit specific treatment in the literature. On the other hand, we hope that (failings of) our approach may point at graphs which require us to identify new lower bound arguments. Overall, we see that already with only small values of $d, m, \mathfrak{r}$, our approach generates proofs for nearly 4/5 of all $G^7 \square P_n$ graph classes. Again, see the appendix for more details.

6 Conclusions

The potency of the approach described in this manuscript is demonstrated by the number of new results that have been established, particularly given that this field of research has been saturated for decades with papers painstakingly establishing one or two results at a time.

In addition to the volume of new results, another benefit of this approach is that it lets us identify the truly difficult graphs, compared to graphs which just require more exhaustive versions of existing arguments. These difficult graphs can then be given individual attention. As shown in the previous section, our approach was successful for all but five graphs from the six-vertex set. Four of these five graphs were previously handled in the literature. Two of them ($i = 31, 156$) required specialised arguments applicable only to those graphs, while the other two ($i = 48, 73$) were corollaries. Researchers in this field can now focus on the one remaining case ($i = 142$), potentially handling not only this graph, but perhaps a family of graphs which emerge from complete graphs with pendant edges added.

Our star-forces, as strengthenings of our basic forces, have essentially accounted for cases where one crossing was able to "hide" in a nearby crossing band. Although this proved to be powerful, it stands to reason that there could be instances where multiple crossings are able to hide, and an accordingly stronger definition and theorem statement would be required. Indeed, an analysis of the $i = 31$ instance shows this to be exactly the case; in this graph, two crossings can hide. Given this context, our approach could perhaps be viewed as the first step in a more general approach that seeks to account for cases where arbitrarily many crossings can hide. Furthermore, it seems likely that the BCCNP framework can be used to attack other families of graphs, particularly those resulting from graph products. In particular, we anticipate that it should be possible to obtain new results for Cartesian products with cycles, although this will likely require further fine-tuning of the crossing bands and corresponding forces. Other common families such as Cartesian products with stars, as well as other kinds of graph products such as strong products, should also be considered in this framework.

References

1 Bernado M. Ábrego, Oswin Aichholzer, Silvia Fernández-Merchant, Thomas Hackl, Jürgen Pammer, Alexander Pilz, Pedro Ramos, Gelasio Salazar, and Birgit Vogtunhuber. All Good Drawings of Small Complete Graphs. In *Proc. 31st European Workshop on Computational Geometry (EuroCG)*, pages 57–60, 2015.

2 Jay Adamsson and R. Bruce Richter. Arrangements, circular arrangements and the crossing number of $C_7 \times C_n$. *J. Comb. Theory B*, 90(1):21–39, 2004. doi:10.1016/j.jctb.2003.05.001.

3 Mark Anderson, R. Bruce Richter, and Peter Rodney. The Crossing Number of $C_6 \times C_6$. *Congressus Numerantium*, 118:97–108, 1996.

4 Mark Anderson, R. Bruce Richter, and Peter Rodney. The Crossing Number of $C_7 \times C_7$. *Congressus Numerantium*, 125:97–117, 1997.

5 Zayed Asiri, Ryan Burdett, Markus Chimani, Michael Haythorpe, Alex Newcombe, and Mirko H. Wagner. Proof files and modified validator. https://tcs.uos.de/research/cr, 2025.

6 Lowell W. Beineke and Richard D. Ringeisen. On the crossing numbers of products of cycles and graphs of order four. *J. Graph Theory*, 4(2):145–155, 1980. doi:10.1002/jgt.3190040203.

7 Drago Bokal. On the crossing numbers of Cartesian products with paths. *J. Comb. Theory B*, 97(3):381–384, 2007. doi:10.1016/J.JCTB.2006.06.003.

8 Drago Bokal, Zdenek Dvorák, Petr Hlinený, Jesús Leaños, Bojan Mohar, and Tilo Wiedera. Bounded degree conjecture holds precisely for c-crossing-critical graphs with $c \leq 12$. *Comb.*, 42(5):701–728, 2022. doi:10.1007/s00493-021-4285-3.

9 Drago Bokal and Jesús Leaños. Characterizing all graphs with 2-exceptional edges. *Ars Math. Contemp.*, 15(2):383–406, 2018. doi:10.26493/1855-3974.1282.378.

10 Christoph Buchheim, Markus Chimani, Dietmar Ebner, Carsten Gutwenger, Michael Jünger, Gunnar W. Klau, Petra Mutzel, and René Weiskircher. A branch-and-cut approach to the crossing number problem. *Discret. Optim.*, 5(2):373–388, 2008. doi:10.1016/j.disopt.2007.05.006.

11 Markus Chimani, Carsten Gutwenger, and Petra Mutzel. Experiments on exact crossing minimization using column generation. *ACM J. Exp. Algorithmics*, 14, 2009. doi:10.1145/1498698.1564504.

12 Markus Chimani, Michael Jünger, and Michael Schulz. Crossing Minimization meets Simultaneous Drawing. In *IEEE VGTC Pacific Visualization Symposium 2008, PacificVis 2008, Kyoto, Japan, March 5-7, 2008*, pages 33–40. IEEE Computer Society, 2008. doi:10.1109/PACIFICVIS.2008.4475456.

13 Markus Chimani, Petra Mutzel, and Immanuel M. Bomze. A New Approach to Exact Crossing Minimization. In Dan Halperin and Kurt Mehlhorn, editors, *Algorithms - ESA 2008, 16th Annual European Symposium, Karlsruhe, Germany, September 15-17, 2008. Proceedings*, volume 5193 of *Lecture Notes in Computer Science*, pages 284–296. Springer, 2008. doi:10.1007/978-3-540-87744-8_24.

14 Markus Chimani and Tilo Wiedera. An ILP-based Proof System for the Crossing Number Problem. In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, volume 57 of *LIPIcs*, pages 29:1–29:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ESA.2016.29.

15 Kieran Clancy, Michael Haythorpe, and Alex Newcombe. A survey of graphs with known or bounded crossing numbers. *Australas. J Comb.*, 78:209–296, 2020. URL: http://ajc.maths.uq.edu.au/pdf/78/ajc_v78_p209.pdf.

16 Kieran Clancy, Michael Haythorpe, Alex Newcombe, and Ed Pegg. There Are No Cubic Graphs on 26 Vertices with Crossing Number 10 or 11. *Graphs Comb.*, 36(6):1713–1721, 2020. doi:10.1007/s00373-020-02204-6.

17 Alice M. Dean and R. Bruce Richter. The crossing number of $C_4 \times C_4$. *J. Graph Theory*, 19(1):125–129, 1995. doi:10.1002/jgt.3190190113.

18 Michael R. Garey and David S. Johnson. Crossing Number Is NP-complete. *SIAM J. Algebraic Disc. Meth.*, 4(3):312–316, 1983. doi:10.1137/0604033.

19 Lev Yu. Glebsky and Gelasio Salazar. The crossing number of $C_m \times C_n$ is as conjectured for $n \geq m(m+1)$. *J. Graph Theory*, 47(1):53–72, 2004. doi:10.1002/jgt.20016.

20 F. Harary, Paul Kainen, and Allen Schwenk. Toroidal Graphs with Arbitrarily High Crossing Number. *Nanta Mathematica*, 6, 1973.

21 Youngsik Huh and Ryo Nikkuni. Crossing number of graphs and ΔY-move, 2024. arXiv:2402.10633.

22 Stanislav Jendrol' and Mária Ščerbová. On the crossing numbers of $S_m \times P_n$ and $S_m \times C_n$. *Časopis pro pěstování matematiky*, 107(3):225–230, 1982. doi:10.21136/CPM.1982.118128.

23 Marián Klešč. On the Crossing Numbers of Cartesian Products of Stars and Paths or Cycles. *Math. Slovaca*, 41(2):113–120, 1991. URL: http://eudml.org/doc/34313.

24 Marián Klešč. The crossing numbers of products of paths and stars with 4-vertex graphs. *Journal of Graph Theory*, 18(6):605–614, 1994.

25 Marián Klešč. The crossing numbers of certain Cartesian products. *Discuss. Math. Graph Theory*, 15(1):5–10, 1995. doi:10.7151/dmgt.1001.

26 Marián Klešč. The crossing number of $K_{2,3} \times P_n$ and $K_{2,3} \times S_n$. *Tatra Moutains Math. Publ*, 9:51–56, 1996.

27 Marián Klešč. The crossing number of $K_5 \times P_n$. *Tatra Moutains Math. Publ*, 18:63–68, 1999.

28 Marián Klešč. The crossing numbers of products of a 5-vertex graph with paths and cycles. *Discuss. Math. Graph Theory*, 19(1):59–69, 1999. doi:10.7151/dmgt.1085.

29 Marián Klešč. The crossing numbers of Cartesian products of paths with 5-vertex graphs. *Discret. Math.*, 233(1-3):353–359, 2001. doi:10.1016/S0012-365X(00)00251-X.

30 Marián Klešč and Jana Petrillová. The crossing numbers of products of path with graphs of order six. *Discuss. Math. Graph Theory*, 33(3):571–582, 2013. doi:10.7151/dmgt.1684.

31 Marián Klešč, Jana Petrillová, and Matús Valo. On the crossing numbers of Cartesian products of wheels and trees. *Discuss. Math. Graph Theory*, 37(2):399–413, 2017. doi:10.7151/dmgt.1957.

32 Marián Klešč, R. Bruce Richter, and Ian Stobert. The crossing number of $C_5 \times C_n$. *J. Graph Theory*, 22(3):239–243, 1996. doi:10.1002/(SICI)1097-0118(199607)22:3\%3C239::AID-JGT4\%3E3.0.CO;2-N.

33 Dan McQuillan, Shengjun Pan, and R. Bruce Richter. On the crossing number of K_{13}. *J. Comb. Theory B*, 115:224–235, 2015. doi:10.1016/j.jctb.2015.06.002.

34 Jana Petrillova. On the optimal drawings of Cartesian products of special 6-vertex graphs with path. *Mathematical Modelling and Geometry*, 3(3):19–28, 2015. URL: http://mmg.tversu.ru/images/publications/2015-vol3-n3/Petrilova-2015-12-02.pdf.

35 R. Bruce Richter and Gelasio Salazar. The crossing number of $C_6 \times C_n$. *Australas. J Comb.*, 23:135–144, 2001. URL: http://ajc.maths.uq.edu.au/pdf/23/ocr-ajc-v23-p135.pdf.

36 R. Bruce Richter and Carsten Thomassen. Intersections of curve systems and the crossing number of $C_5 \times C_5$. *Discret. Comput. Geom.*, 13:149–159, 1995. doi:10.1007/BF02574034.

37 Marcus Schaefer. The Graph Crossing Number and Its Variants: A Survey. *Elec. J. Combin.*, DS21 version 4, 2020. doi:10.37236/2713.

38 Michal Stas. On the Crossing Numbers of the Joining of a Specific Graph on Six Vertices with the Discrete Graph. *Symmetry*, 12(1):135, 2020. doi:10.3390/sym12010135.

39 Michal Staš and Mária Timková. The crossing numbers of join products of four graphs of order five with paths and cycles. *Opuscula Mathematica*, 43(6), 2023. doi:10.7494/OpMath.2023.43.6.865.

40 Michal Staš. Join Products $K_{2,3} + C_n$. *Mathematics*, 8(6):925, 2020. doi:10.3390/math8060925.

41 Michal Staš. The crossing numbers of join products of paths and cycles with four graphs of order five. *Mathematics*, 9(11):1277, 2021.

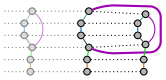
42 Douglas R. Woodall. Cyclic-order graphs and Zarankiewicz's crossing-number conjecture. *J. Graph Theory*, 17(6):657–671, 1993. doi:10.1002/jgt.3190170602.

43 Wenping Zheng, Xiaohui Lin, Yuansheng Yang, and Chong Cui. On the crossing number of $K_m \square P_n$. *Graphs Comb.*, 23(3):327–336, 2007. doi:10.1007/s00373-007-0726-z.

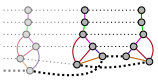
A APPENDIX

Figures A1–A3 list the smallest (d, m) pair for which at least one of Theorems 4 and 6 establishes the lower bounds for each successfully solved graph. The background color intensity reflects how high the values are. The $\mathtt{d,m}^*$ indicates that Theorem 4 does not suffice and that Theorem 6 with $\mathfrak{r} = 0$ is needed. $\mathtt{d,m}^*_*$ indicates that Theorem 6 with $\mathfrak{r} = d - m + 1$ is needed. The blueness of the background also reflects this. Proof files for each of the obtained values can be found at [5] along with a tool that can be used to verify the results.

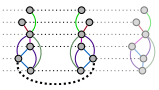
Then, all that remains is to handle the upper bounds. Also contained in Figures A1–A3 are small figures corresponding to each graph, indicating a drawing procedure which establishes the desired upper bound. The small figures should be interpreted as follows. To draw $G \square P_n$ with the indicated number of crossings, $n + 1$ copies of G should be drawn side by side, with the path edges forming horizontal lines. We call the first and last copies of G the *end copies*, and the remaining $n - 1$ copies are the *middle copies*. In the small figures from Figure A2, we show two drawings of G. The end copies should be drawn identically to the right-most drawing in the small figure (taking the mirror image for the first copy). All of the middle copies should be drawn identically to the left-most drawing in the small figure. The edge colors are chosen arbitrarily, but are consistent between the copies. Any edges which end up stubs at the top and bottom are routed around the graph and connected according to the symbol of the stub and the edge color. The figures are presented in such a way that routing the edges around the graph never needs to introduce any additional crossings. If, in the small figure, the left-most drawing contains c_l crossings, and the right-most drawing contains c_r crossings, then an upper bound of $c_l(n-1) + 2c_r$ is established for $\mathrm{cr}(G \square P_n)$. For the graphs marked by a ! between its two copies in Figures A2 and A3, we need to draw two neighboring copies together either at the ends or in the middle for a sufficient upper bound. Essentially, we use one (or more) of the following three strategies:



Two last copies (G^6_{42})



Two last copies (G^6_{63})



Two middle copies (G^6_{82})

We use the following symbols to indicate the status of the results:

symbol	previously known	solved by us
✚	no	yes
✚	not peer-reviewed	yes
✔	yes	yes
✔	yes	no
✘	no	no

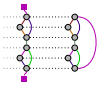
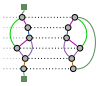
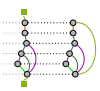
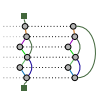
Figure A1 Results for 5-vertex graphs. We omit the planar instances $i = 1, 8$.

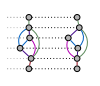
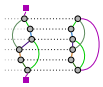
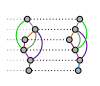
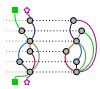
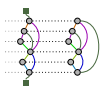
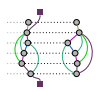
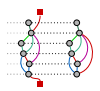
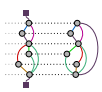
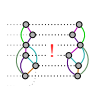
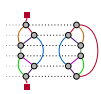
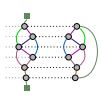
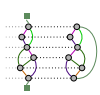
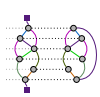
i	G_i^5	cr d,m
2 ✔		$2n-2$ 3,1*
3 ✔		$1n-1$ 2,1
4 ✔		$1n-1$ 2,1
5 ✔		$1n-1$ 2,1
6 ✔		$2n-2$ 3,1*
7 ✔		$1n-1$ 2,1
9 ✔		$2n-2$ 2,1
10 ✔		$2n+0$ 2,1
11 ✔		$2n-2$ 2,1
12 ✔		$2n-2$ 2,1
13 ✔		$1n-1$ 2,1
14 ✔		$2n-2$ 2,1
15 ✔		$3n-1$ 2,1
16 ✔		$3n-1$ 2,1
17 ✔		$2n+0$ 2,1
18 ✔		$3n-1$ 2,1
19 ✔		$3n-1$ 2,1
20 ✔		$4n+0$ 2,1
21 ✔		$6n+0$ 2,1

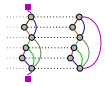
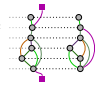
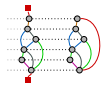
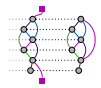
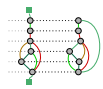
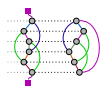
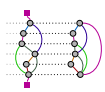
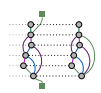
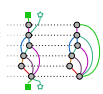
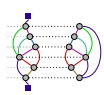
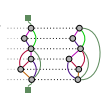
Figure A2 Results for 6-vertex graphs.

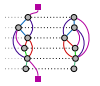
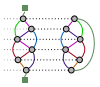
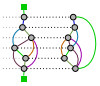
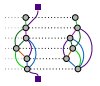
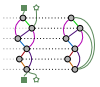
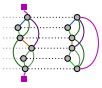
i	G_i^6	cr d,m
25 ✔		$0n+0$ 2,1
26 ✔		$1n-1$ 2,1
27 ✔		$2n-2$ 3,1*
28 ✔		$1n-1$ 2,1
29 ✔		$2n-2$ 3,1*
31 ✓		$4n-4$
40 ✔		$0n+0$ 2,1
41 ✔		$1n-1$ 2,1
42 ✔		$2n-4$ 4,2
43 ✔		$1n-1$ 2,1
44 ✔		$2n-2$ 2,1
45 ✔		$2n-2$ 2,1
46 ✔		$1n-1$ 2,1
47 ✔		$2n-2$ 3,1*
48 ✓		$4n-4$
49 ✢		$2n-2$ 2,1
51 ✔		$3n-3$ 3,1*
53 ✔		$2n-2$ 3,1*
54 ✔		$2n-2$ 2,1
59 ✔		$2n-2$ 2,1
60 ✔		$1n-1$ 2,1
61 ✔		$2n+0$ 2,1
62 ✔		$3n-5$ 4,2*
63 ✔		$2n-2$ 2,1
64 ✔		$2n-2$ 2,1
65 ✔		$3n-3$ 2,1
66 ✔		$2n-2$ 2,1
67 ✢		$3n-3$ 3,1

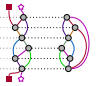
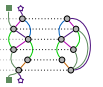
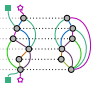
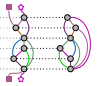
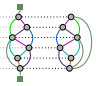
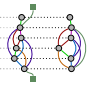
i	G_i^6	cr d,m
68 ✔		$3n-1$ 2,1
70 ✔		$3n-3$ 3,1
71 ✔		$3n-1$ 2,1
72 ✔		$4n-4$ 3,1*
73 ✓		$4n-4$
74 ✔		$2n-2$ 2,1
75 ✔		$2n+0$ $3,1^*_*$
76 ✚		$3n-3$ 2,1
77 ✔		$2n-2$ 3,1*
78 ✚		$3n-3$ 3,1*
79 ✔		$4n-4$ 3,1*
80 ✔		$4n-4$ 2,1*
81 ✚		$5n-3$ 3,1
82 ✚		$4n-2$ 3,1*

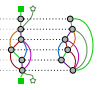
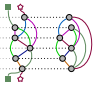
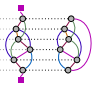
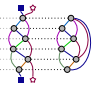
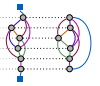
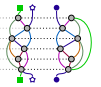
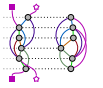
i	G_i^6	cr d,m
83 ✔		$2n-2$ 2,1
84 ✚		$3n-1$ 2,1
85 ✔		$2n+0$ 2,1
86 ✔		$3n-1$ 2,1
87 ✔		$3n-1$ $2,1^*_*$
88 ✚		$4n-2$ 3,1
89 ✔		$3n-3$ 2,1
90 ✔		$3n-3$ 2,1
91 ✔		$3n-1$ 2,1
92 ✚		$3n-3$ 2,1
93 ✚		$4n+0$ 2,1
94 ✔		$2n-2$ 2,1
95 ✚		$3n-1$ 2,1
98 ✚		$3n-1$ $3,1^*_*$

i	G_i^6	cr d,m
99 ✚		$4n-2$ 3,1
100 ✚		$4n-2$ 2,1
101 ✚		$4n-2$ 2,1
102 ✚		$5n-3$ 3,1*
103 ✔		$6n-2$ $2,1^*_*$
104 ✔		$4n-4$ 2,1*
105 ✚		$6n-2$ $3,1^*_*$
106 ✚		$5n-3$ 3,1
107 ✚		$5n-3$ $2,1^*_*$
108 ✚		$4n-2$ 2,1
109 ✚		$4n+0$ 2,1
110 ✚		$3n-1$ 2,1
111 ✔		$3n-1$ 2,1
112 ✚		$4n+0$ 2,1

i	G_i^6	cr d,m
113 ✔		$4n-4$ 2,1
114 ✚		$5n-3$ 3,1
115 ✚		$4n-2$ 2,1
116 ✚		$4n-2$ 2,1
118 ✚		$4n-2$ 2,1
119 ✚		$7n-1$ $2,1^*_*$
120 ✔		$3n-3$ 2,1
121 ✚		$4n+0$ 2,1
122 ✚		$5n-3$ 2,1
123 ✚		$5n-3$ 3,1
124 ✚		$5n-3$ 2,1
125 ✔		$5n-3$ 2,1
126 ✚		$6n-2$ $2,1^*_*$
127 ✚		$5n-3$ $2,1^*_*$

i	G_i^6	cr d,m
128		$6n-2$ 2,1*
129		$6n-2$ 3,1
130		$4n+0$ 2,1
131		$5n-1$ 2,1
132		$6n-2$ 3,1
133		$4n-2$ 2,1
134		$6n-4$ 3,1

i	G_i^6	cr d,m
135		$7n-1$ 2,1
137		$4n+0$ 2,1
138		$5n-1$ 2,1
139		$7n-1$ 2,1*
140		$6n-2$ 2,1*
141		$6n-2$ 2,1
142		$9n-3$

i	G_i^6	cr d,m
143		$6n-2$ 2,1
144		$8n-2$ 2,1
145		$7n-1$ 2,1
146		$5n-1$ 2,1
147		$8n-2$ 3,1
148		$7n-1$ 2,1
149		$10n+0$ 2,1*

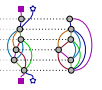
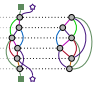
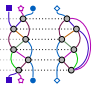
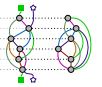
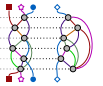
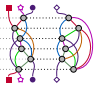
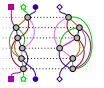
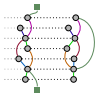
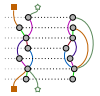
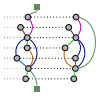
i	G_i^6	cr d,m
150		$9n-3$ 2,1
151		$8n-2$ 2,1
152		$6n+0$ 2,1
153		$10n+0$ 2,1
154		$9n-1$ 2,1
155		$12n+0$ 2,1*
156		$15n+3$

Figure A3 Successful results for 7-vertex graphs.

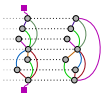
i	G_i^7	cr d,m
4		$2n-2$ 3,1*
6		$4n-4$ 3,1*
9		$2n-2$ 3,1*

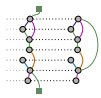
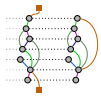
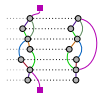
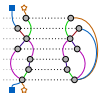
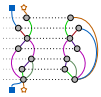
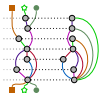
i	G_i^7	cr d,m
10		$3n-3$ 2,1
11		$3n-3$ 3,1*
12		$5n-3$ 3,1*

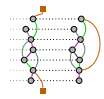
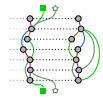
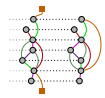
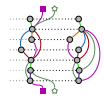
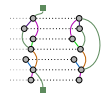
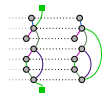
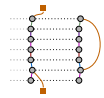
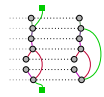
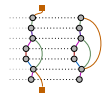
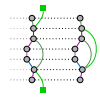
i	G_i^7	cr d,m
15		$4n-2$ 2,1*
16		$6n-2$ 2,1*
17		$6n-4$ 3,1*

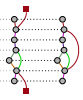
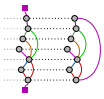
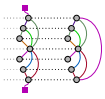
i	G_i^7	cr d,m
21		$2n-2$ 3,1*
24		$1n-1$ 2,1
25		$3n-3$ 3,1

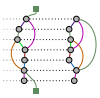
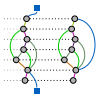
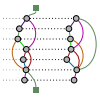
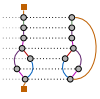
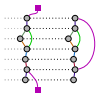
i	G_i^7	cr d,m
26		$2n-2$ 2,1
27		$2n-2$ 2,1
29		$4n-4$ 3,1
30		$4n-4$ 2,1
32		$1n-1$ 2,1
33		$2n-2$ 3,1*
34		$3n-5$ 4,2*
35		$2n-2$ 3,1*
36		$2n-2$ 2,1
38		$4n-4$ 3,1*
45		$2n-2$ 2,1
46		$4n-2$ 2,1*
47		$3n-3$ 2,1

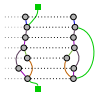
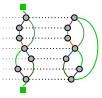
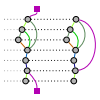
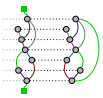
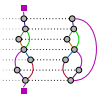
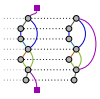
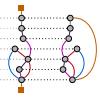
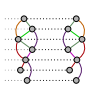
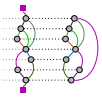
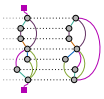
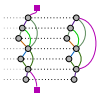
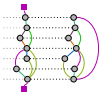
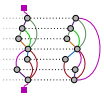
i	G_i^7	cr d,m
48		$3n-1$ 2,1
49		$4n-4$ 3,1*
50		$4n-4$ 3,1*
51		$3n-3$ 2,1
52		$5n-3$ 2,1*
53		$6n-4$ 3,1*
54		$5n-5$ 3,1*
55		$5n-3$ 3,1*
56		$7n-5$ 4,2*
57		$2n-1$ 3,1*
58		$2n-2$ 3,1*
59		$4n-4$ 4,2
60		$3n-1$ 3,1*

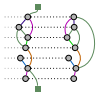
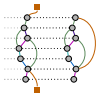
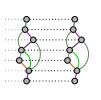
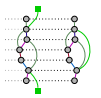
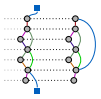
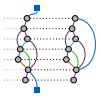
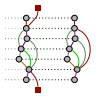
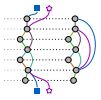
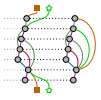
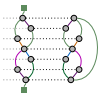
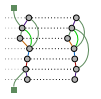
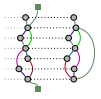
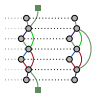
i	G_i^7	cr d,m
62		$3n-3$ 3,1*
63		$5n-3$ 3,1*
65		$5n-5$ 3,1*
67		$3n-3$ 2,1
68		$3n-1$ 3,1*
69		$5n-3$ 3,1*
71		$5n-3$ 4,2*
73		$6n-2$ 2,1*
74		$4n-2$ 2,1*
75		$5n-1$ 2,1*
76		$6n-2$ 2,1*
78		$6n-4$ 2,1*
80		$8n-4$ 3,1*

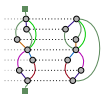
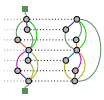
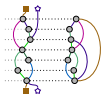
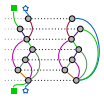
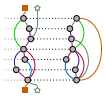
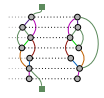
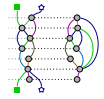
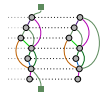
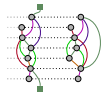
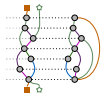
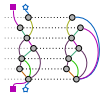
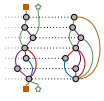
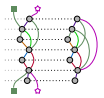
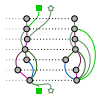
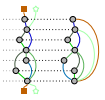
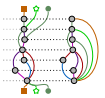
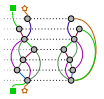
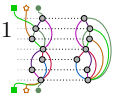
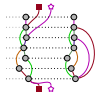
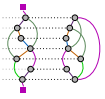
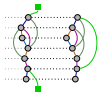
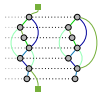
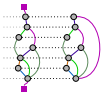
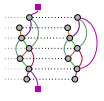
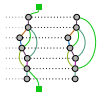
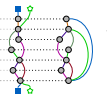
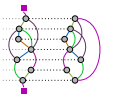
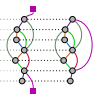
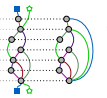
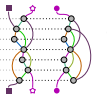
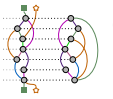
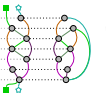
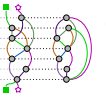
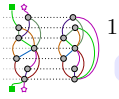
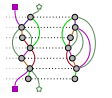
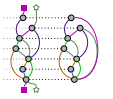
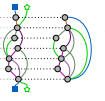
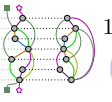
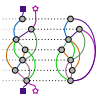
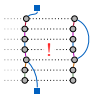
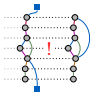
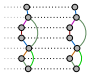
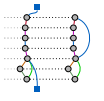
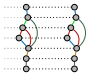
i	G_i^7	cr d,m
81		$4n-2$ 2,1
82		$6n-2$ 2,1*
83		$6n-4$ 3,1*
84		$8n-4$ 4,2*
85		$8n+0$ 2,1*
87		$4n-2$ 2,1*
88		$4n-4$ 3,1*
91		$1n-1$ 2,1
92		$2n-2$ 2,1
93		$3n-3$ 2,1
94		$3n-3$ 2,1
95		$3n-3$ 2,1
96		$4n-4$ 3,1

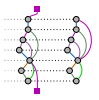
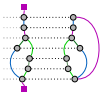
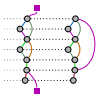
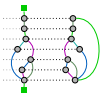
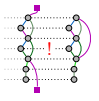
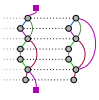
i	G_i^7	cr d,m
97		$2n-2$ 2,1
98		$4n-2$ $3,1^*_*$
99		$3n-3$ 2,1
100		$4n-4$ 2,1
101		$5n-5$ 3,1*
102		$4n-4$ 2,1
103		$6n-6$ 3,1*
104		$2n-4$ 4,2
105		$1n-1$ 2,1
107		$2n-2$ 2,1
108		$4n-4$ 3,1*
109		$3n-5$ 4,2*
110		$3n-3$ 3,1

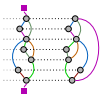
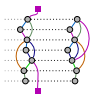
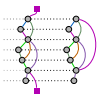
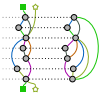
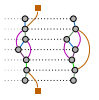
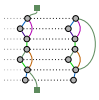
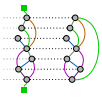
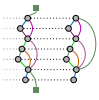
i	G_i^7	cr d,m
112		$2n-2$ 2,1
114		$4n-4$ 3,1*
116		$4n-2$ $2,1^*_*$
117		$5n-3$ 2,1
118		$5n-3$ $3,1^*_*$
119		$4n-2$ 2,1
120		$6n-4$ $3,1^*_*$
121		$6n-2$ 4,2
122		$6n-4$ $2,1^*_*$
124		$6n-4$ $3,1^*_*$
125		$8n-4$ $4,2^*_*$
126		$2n+0$ 2,1
127		$2n-2$ 2,1

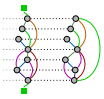
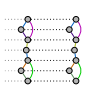
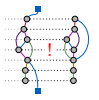
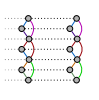
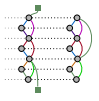
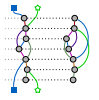
i	G_i^7	cr d,m
128		$3n-1$ 2,1
129		$4n-2$ $2,1^*_*$
130		$4n-4$ 3,1
131		$5n-3$ $3,1^*_*$
132		$3n-1$ 2,1
133		$4n-4$ 2,1
135		$4n-2$ $3,1^*_*$
136		$4n-2$ $3,1^*_*$
137		$5n-3$ $2,1^*_*$
138		$6n-4$ $3,1^*_*$
139		$5n-5$ 2,1*
141		$5n-3$ $2,1^*_*$
142		$7n-5$ $3,1^*_*$

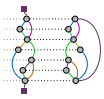
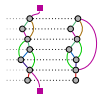
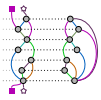
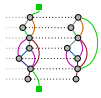
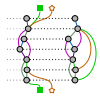
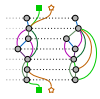
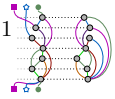
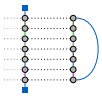
i	G_i^7	cr d,m
143		$4n-2$ 2,1
147		$4n-2$ 2,1
148		$4n-2$ $4,2^*_*$
149		$5n-3$ $3,1^*_*$
150		$3n-3$ 2,1
152		$5n-3$ $2,1^*_*$
153		$6n-4$ $3,1^*_*$
154		$5n-5$ 2,1
155		$7n-5$ $2,1^*_*$
156		$4n+0$ 2,1
157		$5n-3$ $3,1^*_*$
158		$4n-2$ $2,1^*_*$
160		$4n-4$ 2,1

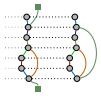
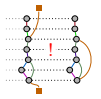
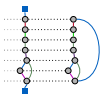
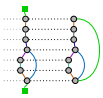
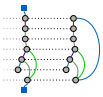
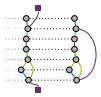
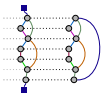
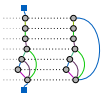
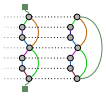
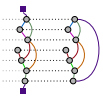
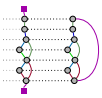
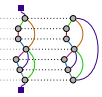
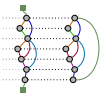
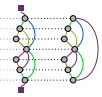
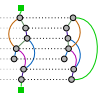
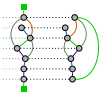
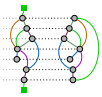
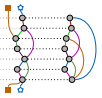
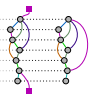
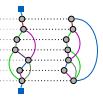
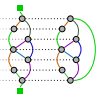
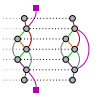
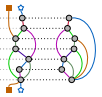
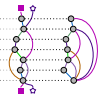
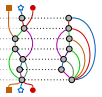
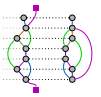
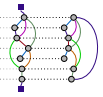
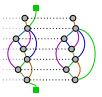
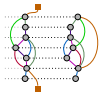
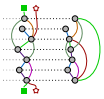
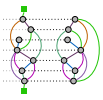
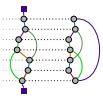
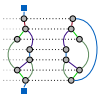
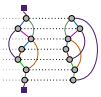
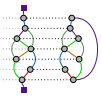
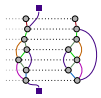
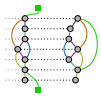
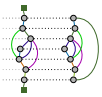
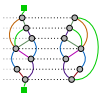
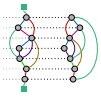
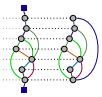
i	G_i^7	cr d,m
161		$6n-2$ 2,1$^*_*$
162		$6n-4$ 3,1$^*_*$
163		$8n-4$ 4,2$^*_*$
164		$6n-2$ 2,1$^*_*$
165		$7n-1$ 2,1$^*_*$
166		$8n-4$ 3,1$^*_*$
168		$5n-1$ 4,2
169		$7n-3$ 4,2
170		$7n-3$ 3,1
171		$8n-4$ 4,2$^*_*$
173		$5n-1$ 4,2
174		$6n-2$ 3,1$^*_*$
175		$7n-3$ 3,1$^*_*$
176		$7n-5$ 3,1$^*_*$
177		$8n-4$ 4,2$^*_*$
178		$6n-4$ 2,1$^*_*$
179		$8n-4$ 3,1$^*_*$
180		$8n+0$ 2,1$^*_*$
181		$10n-2$ 3,1$^*_*$
183		$6n-2$ 2,1$^*_*$
187		$6n-2$ 2,1
188		$6n-4$ 3,1$^*_*$
189		$6n-2$ 3,1$^*_*$
190		$5n-3$ 2,1$^*_*$
191		$6n-4$ 3,1$^*_*$
192		$5n-3$ 3,1$^*_*$
193		$4n-2$ 2,1$^*_*$
194		$7n-3$ 4,2
195		$7n-3$ 4,2
196		$6n-4$ 2,1$^*_*$
197		$8n-4$ 4,2
198		$9n-1$ 2,1$^*_*$
200		$6n+0$ 2,1
201		$9n-1$ 2,1$^*_*$
203		$11n-3$ 2,1$^*_*$
205		$9n-3$ 4,2$^*_*$
207		$8n-2$ 2,1$^*_*$
208		$8n-4$ 4,2
209		$10n-2$ 3,1$^*_*$
213		$11n-1$ 2,1$^*_*$
220		$0n+0$ 2,1
221		$2n-2$ 2,1
222		$1n-1$ 2,1
223		$2n-4$ 4,2
224		$2n-2$ 3,1*
225		$1n-1$ 2,1
226		$3n-3$ 3,1*
227		$3n-5$ 4,2
228		$3n-3$ 3,1*
229		$2n-2$ 2,1
230		$3n-3$ 3,1
231		$3n-3$ 2,1

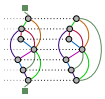
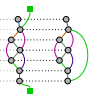
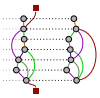
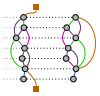
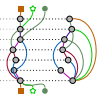
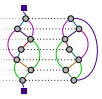
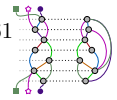
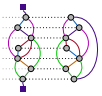
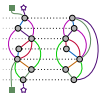
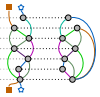
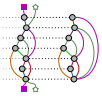
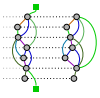
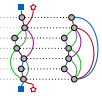
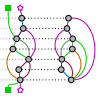
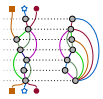
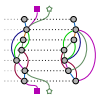
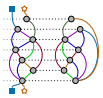
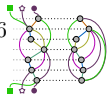
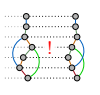
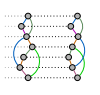
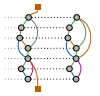
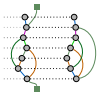
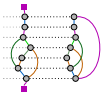
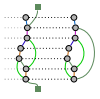
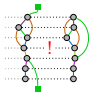
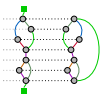
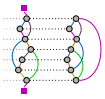
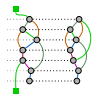
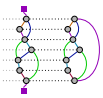
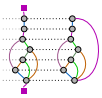
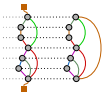
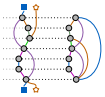
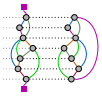
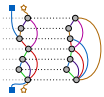
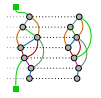
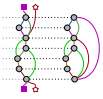
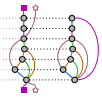
i	G_i^7	cr d,m
233		$4n-4$ 3,1
235		$4n-4$ 2,1
236		$6n-6$ 3,1*
237		$3n-3$ 3,1*
240		$4n-2$ 3,1*
242		$3n-1$ 2,1
243		$5n-3$ 3,1*
244		$3n-3$ 2,1
245		$4n-2$ 2,1*
246		$5n-3$ 3,1*
248		$4n-6$ 4,2*
249		$5n-3$ 3,1*
250		$4n-4$ 3,1

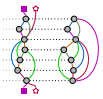
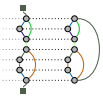
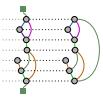
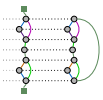
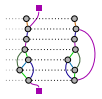
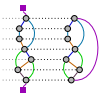
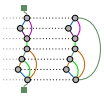
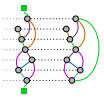
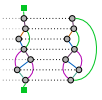
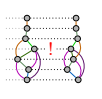
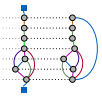
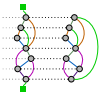
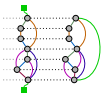
i	G_i^7	cr d,m
251		$5n-3$ 3,1*
252		$5n-3$ 2,1*
253		$6n-4$ 3,1*
254		$5n-5$ 2,1
255		$6n-4$ 3,1*
256		$7n-5$ 2,1*
257		$3n-3$ 2,1
258		$4n-2$ 2,1
259		$3n-3$ 2,1
260		$3n-1$ 2,1
261		$5n-3$ 2,1*
262		$6n-4$ 3,1*
263		$5n-5$ 2,1

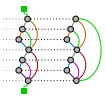
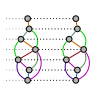
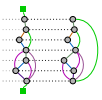
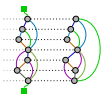
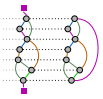
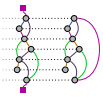
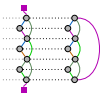
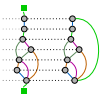
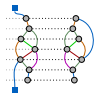
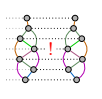
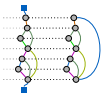
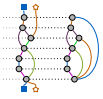
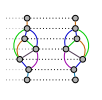
i	G_i^7	cr d,m
264		$5n-3$ 3,1*
265		$7n-5$ 3,1*
266		$2n-2$ 3,1*
267		$4n-4$ 4,2
268		$4n-4$ 3,1
269		$3n-1$ 3,1*
270		$4n-2$ 2,1*
271		$4n-4$ 3,1*
272		$3n-3$ 3,1*
273		$5n-3$ 3,1*
275		$5n-5$ 3,1*
276		$5n-3$ 4,2*
278		$6n-2$ 2,1*

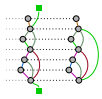
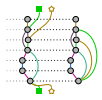
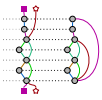
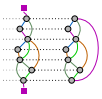
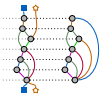
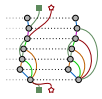
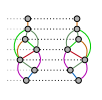
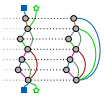
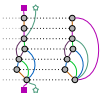
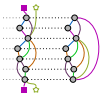
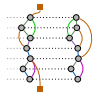
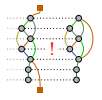
i	G_i^7	cr d,m
279		$5n-3$ 2,1*
280		$4n-2$ 2,1
281		$5n-1$ 3,1*
282		$6n-2$ 4,2
284		$7n-5$ 3,1*
285		$6n-4$ 2,1*
286		$7n-3$ 3,1*
287		$8n-4$ 3,1*
288		$6n-2$ 3,1*
289		$8n-4$ 4,2
290		$8n+0$ 2,1*
291		$10n-2$ 3,1*
292		$0n+0$ 2,1

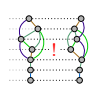
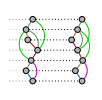
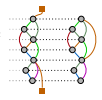
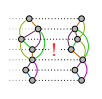
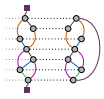
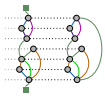
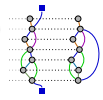
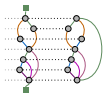
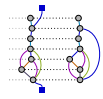
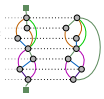
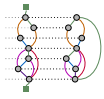
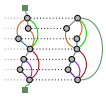
i	G_i^7	cr d,m
293		$3n-3$ 2,1
294		$3n-5$ 4,2
295		$1n-1$ 2,1
296		$2n-2$ 2,1
297		$2n+0$ 3,1$^*_*$
298		$3n-3$ 3,1
299		$3n-2$ 3,1$^*_*$
300		$3n-1$ 3,1$^*_*$
301		$2n-2$ 2,1
302		$3n-3$ 2,1
303		$4n-2$ 3,1$^*_*$
304		$3n-1$ 3,1$^*_*$
305		$3n-3$ 2,1
306		$4n-4$ 2,1
307		$4n-4$ 3,1
308		$4n-4$ 3,1
309		$2n-2$ 2,1
310		$3n-1$ 3,1$^*_*$
311		$5n-5$ 3,1
312		$4n-4$ 2,1
313		$5n-3$ 3,1$^*_*$
314		$6n-6$ 2,1
315		$4n-2$ 3,1$^*_*$
316		$5n-3$ 3,1$^*_*$
317		$4n-2$ 3,1$^*_*$
318		$6n-2$ 4,2$^*_*$
320		$5n-3$ 2,1
321		$5n-1$ 3,1$^*_*$
322		$5n-3$ 3,1$^*_*$
323		$5n-3$ 2,1$^*_*$
324		$5n-1$ 2,1
325		$5n-3$ 2,1
326		$7n-3$ 3,1$^*_*$
327		$6n-4$ 2,1$^*_*$
328		$7n-3$ 4,2
329		$8n-4$ 3,1$^*_*$
330		$5n-1$ 2,1
331		$6n-4$ 3,1$^*_*$
332		$7n-3$ 4,2$^*_*$
333		$7n-3$ 4,2
334		$7n-3$ 4,2
335		$9n-3$ 4,2$^*_*$
336		$4n-2$ 2,1
337		$4n+0$ 2,1
338		$5n-1$ 2,1$^*_*$
339		$4n+0$ 2,1
340		$4n-2$ 2,1
341		$5n-3$ 3,1$^*_*$
342		$5n-3$ 2,1
343		$6n-2$ 4,2
344		$6n-2$ 4,2
345		$6n-2$ 2,1$^*_*$

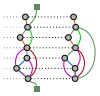
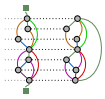
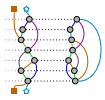
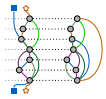
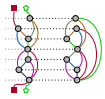
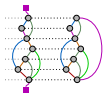
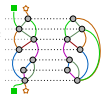
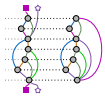
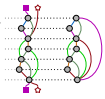
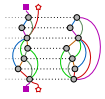
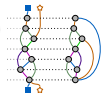
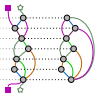
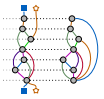
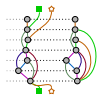
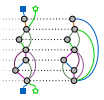
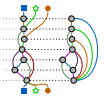
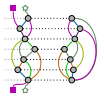
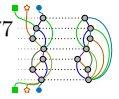
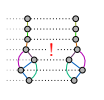
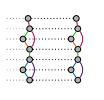
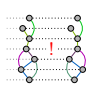
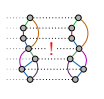
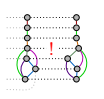
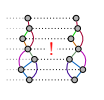
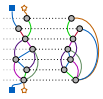
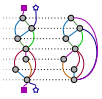
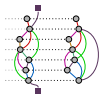
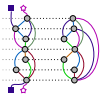
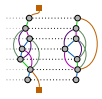
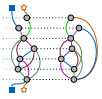
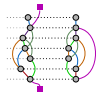
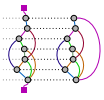
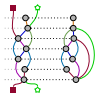
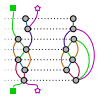
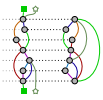
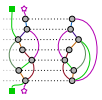
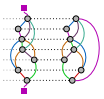
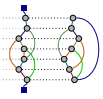
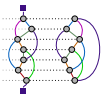
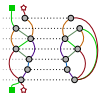
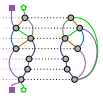
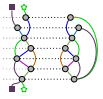
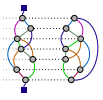
i	G_i^7	cr d,m
346		$6n-4$ $2,1^*_*$
347		$8n-4$ $2,1^*_*$
348		$4n-2$ 2,1
349		$7n-5$ $3,1^*_*$
350		$6n-4$ $3,1^*_*$
351		$6n-4$ $2,1^*_*$
352		$7n-3$ 4,2
353		$7n-5$ $3,1^*_*$
354		$6n-2$ 3,1
355		$8n-4$ 4,2
356		$5n-1$ $3,1^*_*$
357		$7n-3$ 4,2
358		$7n-5$ $2,1^*_*$
359		$9n-3$ $4,2^*_*$
360		$6n+0$ $2,1^*_*$
361		$7n-1$ $2,1^*_*$
362		$8n-2$ $2,1^*_*$
363		$7n-3$ 4,2
364		$7n-1$ $2,1^*_*$
365		$9n-3$ $2,1^*_*$
366		$7n-3$ 4,2
367		$7n-3$ 4,2
368		$9n-3$ $4,2^*_*$
369		$8n-4$ 4,2
370		$7n-5$ $2,1^*_*$
371		$8n-2$ $3,1^*_*$
372		$9n-3$ $2,1^*_*$
373		$7n-3$ 2,1
374		$9n-3$ $2,1^*_*$
375		$9n-1$ 2,1
376		$11n-1$ $3,1^*_*$
377		$4n-4$ 4,2
379		$5n-3$ $3,1^*_*$
381		$4n-2$ $3,1^*_*$
382		$6n-4$ $3,1^*_*$
385		$5n-3$ 2,1
386		$4n-2$ $2,1^*_*$
387		$4n-2$ 2,1
388		$4n-4$ 4,2
389		$3n-1$ 2,1
390		$5n-3$ $2,1^*_*$
391		$6n-2$ 4,2
392		$4n-2$ $2,1^*_*$
393		$5n-1$ 2,1
394		$6n-2$ 4,2
395		$5n-5$ 3,1
396		$6n-4$ $2,1^*_*$
397		$6n-2$ 4,2
398		$7n-5$ 4,2
399		$7n-3$ $3,1^*_*$
400		$6n-4$ $2,1^*_*$
401		$7n-3$ $3,1^*_*$

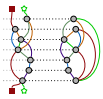
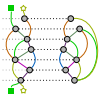
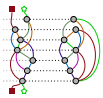
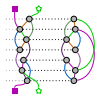
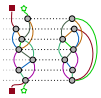
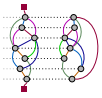
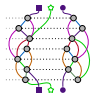
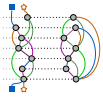
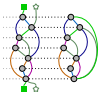
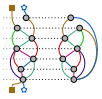
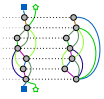
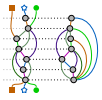
i	G_i^7	cr d,m
402		$8n-4$ 4,2
403		$3n-3$ 2,1
404		$4n-4$ 3,1
405		$2n-2$ 2,1
406		$3n-1$ $2,1^*_*$
407		$4n-2$ 2,1
408		$4n-4$ 2,1
409		$5n-3$ $3,1^*_*$
410		$3n-1$ $2,1^*_*$
411		$5n-3$ $3,1^*_*$
412		$4n-2$ $2,1^*_*$
413		$5n-3$ $2,1^*_*$
414		$6n-4$ $3,1^*_*$

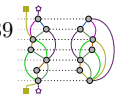
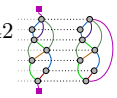
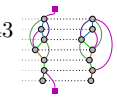
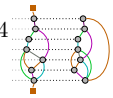
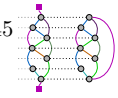
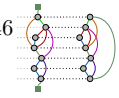
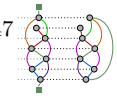
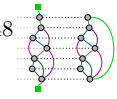
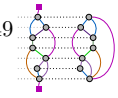
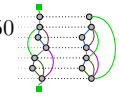
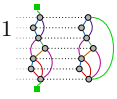
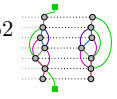
i	G_i^7	cr d,m
415		$6n-6$ 3,1*
416		$6n-2$ $4,2^*_*$
417		$5n-3$ $2,1^*_*$
418		$7n-5$ $3,1^*_*$
419		$4n-2$ 2,1
420		$5n-3$ $3,1^*_*$
421		$3n-3$ 2,1
422		$4n-2$ 2,1
423		$5n-1$ 4,2
424		$4n-2$ $4,2^*_*$
425		$4n-4$ 2,1
426		$5n-3$ $2,1^*_*$
427		$5n-1$ 4,2

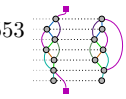
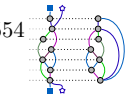
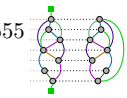
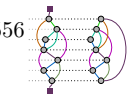
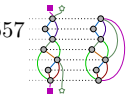
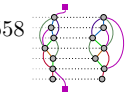
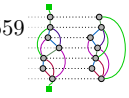
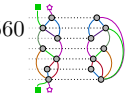
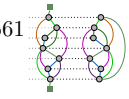
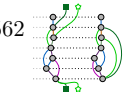
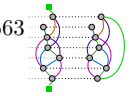
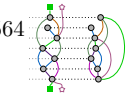
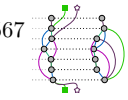
i	G_i^7	cr d,m
428		$5n-5$ 2,1
429		$5n-5$ 3,1*
430		$6n-4$ $3,1^*_*$
431		$4n-4$ 2,1
432		$5n-3$ $2,1^*_*$
433		$6n-4$ $3,1^*_*$
434		$7n-5$ $3,1^*_*$
435		$6n-2$ 4,2
436		$5n-5$ 2,1
437		$6n-4$ $2,1^*_*$
438		$7n-5$ $2,1^*_*$
439		$5n-3$ $3,1^*_*$
440		$5n-5$ 4,2

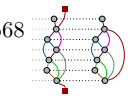
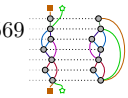
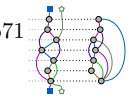
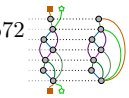
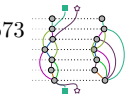
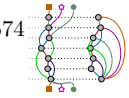
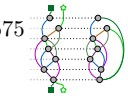
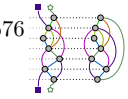
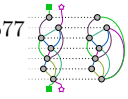
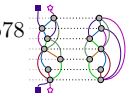
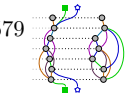
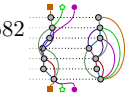
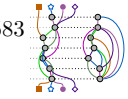
i	G_i^7	cr d,m
441		$4n-2$ $3,1^*_*$
442		$5n-3$ $3,1^*_*$
443		$6n-4$ $3,1^*_*$
444		$6n-4$ $3,1^*_*$
446		$6n-4$ $3,1^*_*$
448		$4n+0$ 2,1
449		$5n-3$ $3,1^*_*$
450		$4n-2$ 2,1
451		$5n-3$ $2,1^*_*$
452		$5n-3$ $3,1^*_*$
454		$6n-2$ $3,1^*_*$
455		$6n-2$ $2,1^*_*$
456		$7n-5$ $3,1^*_*$

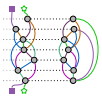
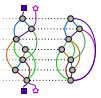
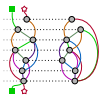
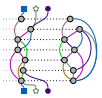
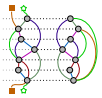
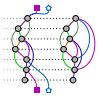
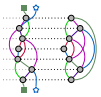
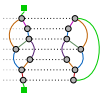
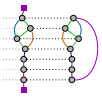
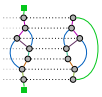
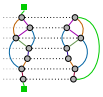
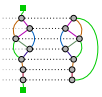
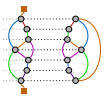
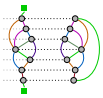
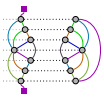
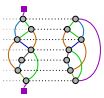
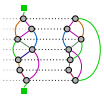
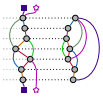
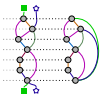
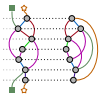
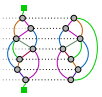
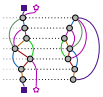
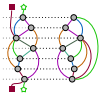
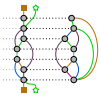
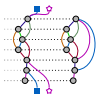
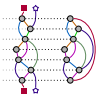
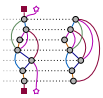
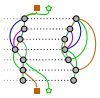
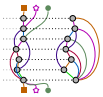
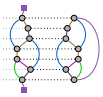
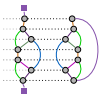
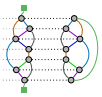
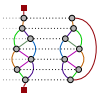
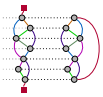
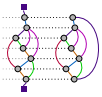
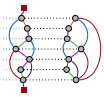
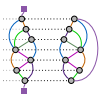
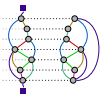
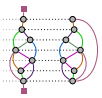
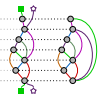
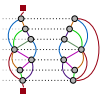
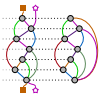
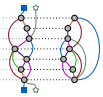
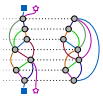
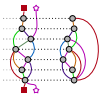
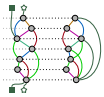
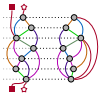
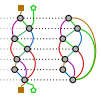
i	G_i^7	cr d,m
457		$6n-4$ 3,1*
458		$8n-4$ 4,2*
459		$6n-2$ 2,1*
460		$7n-1$ 2,1*
461		$8n-4$ 3,1*
462		$9n-3$ 3,1*
463		$6n-2$ 3,1*
464		$7n-3$ 4,2
465		$8n-4$ 4,2*
467		$7n-5$ 2,1*
468		$8n-2$ 2,1*
469		$9n-3$ 4,2
470		$5n-1$ 2,1
471		$6n-2$ 4,2
472		$7n-3$ 3,1*
473		$8n-4$ 4,2
474		$6n-4$ 2,1*
475		$8n-4$ 2,1*
476		$8n+0$ 2,1*
477		$10n-2$ 3,1*
478		$2n-2$ 2,1
479		$2n-2$ 2,1
480		$3n-3$ 3,1*
481		$4n-4$ 3,1*
482		$3n-1$ 2,1*
484		$3n-3$ 2,1
485		$4n-2$ 3,1*
486		$5n-3$ 3,1*
487		$5n-5$ 3,1*
488		$4n-2$ 2,1*
489		$6n-4$ 3,1*
492		$7n-1$ 2,1*
494		$7n-3$ 2,1*
495		$8n-4$ 4,2
496		$9n-3$ 2,1*
497		$9n-3$ 3,1*
500		$10n-2$ 2,1*
503		$8n-2$ 3,1*
504		$8n-2$ 3,1*
505		$7n-3$ 4,2
506		$7n-3$ 3,1*
507		$7n-3$ 2,1*
508		$7n-1$ 2,1*
509		$9n-3$ 3,1*
511		$8n-2$ 4,2
514		$7n-1$ 2,1*
515		$8n-2$ 2,1*
516		$7n-1$ 2,1
517		$6n-2$ 2,1
518		$6n-2$ 3,1
519		$5n-1$ 2,1
520		$8n-2$ 2,1

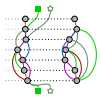
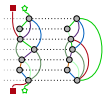
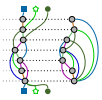
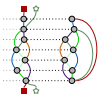
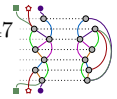
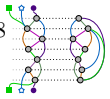
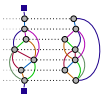
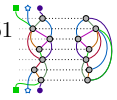
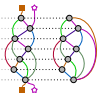
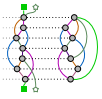
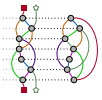
i	G_i^7	cr d,m
521		$9n-3$ 3,1*
522		$7n-3$ 2,1
523		$10n-4$ 2,1*
524		$8n+0$ 2,1
525		$9n-1$ 2,1
526		$10n-1$ 3,1*
527		$10n-2$ 3,1*
528		$11n-3$ 2,1*
529		$10n-4$ 3,1*
534		$8n-2$ 2,1*
535		$8n-2$ 2,1
537		$8n-4$ 4,2
538		$10n-2$ 2,1*

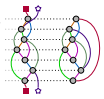
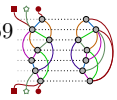
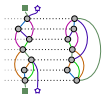
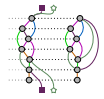
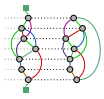
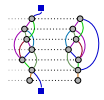
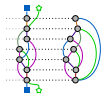
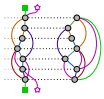
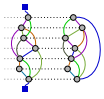
i	G_i^7	cr d,m
539		$9n-3$ 2,1*
541		$13n-1$ 2,1*
542		$6n-2$ 2,1*
543		$6n-2$ 3,1*
544		$6n-2$ 2,1
545		$5n-1$ 2,1
546		$6n-4$ 2,1*
547		$6n-2$ 3,1
548		$5n-3$ 2,1*
549		$6n-2$ 4,2
550		$6n-4$ 3,1*
551		$5n-3$ 2,1*
552		$6n-2$ 3,1

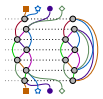
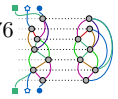
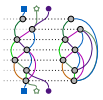
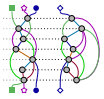
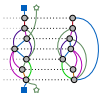
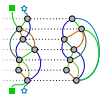
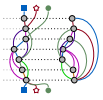
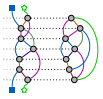
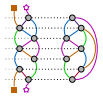
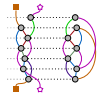
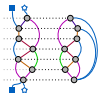
i	G_i^7	cr d,m
553		$5n-1$ 2,1
554		$5n-1$ 3,1*
555		$7n-1$ 2,1*
556		$7n-3$ 2,1
557		$7n-3$ 2,1*
558		$7n-3$ 4,2
559		$7n-3$ 2,1*
560		$7n-3$ 3,1*
561		$9n-3$ 2,1*
562		$6n-2$ 2,1*
563		$9n-3$ 3,1*
564		$8n-2$ 3,1*
567		$7n-3$ 4,2

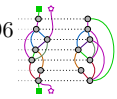
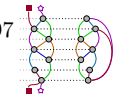
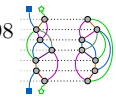
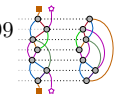
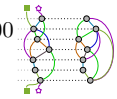
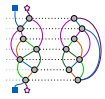
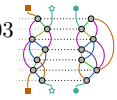
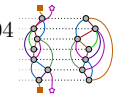
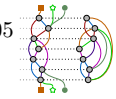
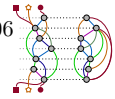
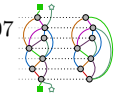
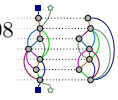
i	G_i^7	cr d,m
568		$6n-4$ 3,1*
569		$4n-2$ 2,1
571		$7n-3$ 2,1*
572		$6n-4$ 2,1
573		$8n-4$ 4,2*
574		$8n-4$ 3,1*
575		$9n-1$ 2,1*
576		$10n-2$ 3,1*
577		$10n-4$ 3,1*
578		$11n-3$ 2,1*
579		$10n-4$ 3,1*
582		$11n-3$ 2,1*
583		$13n-1$ 2,1*

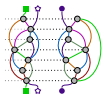
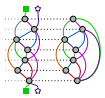
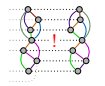
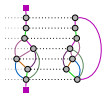
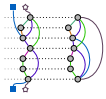
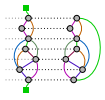
i	G_i^7	cr d,m
584		$12n+0$ 2,1*
585		$11n-1$ 2,1*
586		$11n-1$ 2,1
589		$10n-2$ 2,1*
591		$9n-1$ 2,1*
592		$10n-4$ 3,1*
593		$10n-4$ 3,1*
595		$9n-3$ 3,1
599		$4n-2$ 2,1
600		$2n+0$ 2,1*
601		$3n-1$ 2,1
602		$4n-2$ 2,1*
603		$3n-1$ 2,1
604		$5n-3$ 2,1
605		$5n-3$ 2,1*
606		$7n-5$ 2,1*
607		$6n-2$ 2,1
608		$5n-1$ 2,1
609		$6n+0$ 2,1*
610		$6n-2$ 2,1
611		$5n-3$ 2,1
612		$7n-3$ 3,1*
613		$7n-1$ 2,1
614		$7n-3$ 4,2
615		$9n-3$ 2,1*
616		$5n-1$ 2,1
617		$6n-4$ 4,2
618		$7n-3$ 4,2
619		$8n-4$ 4,2
620		$8n-2$ 2,1*
621		$8n-4$ 4,2
622		$9n-3$ 2,1*
623		$5n-1$ 2,1
624		$4n+0$ 2,1
625		$5n-1$ 2,1
626		$4n+0$ 2,1
627		$4n-2$ 2,1
628		$6n-2$ 2,1
629		$6n-4$ 2,1*
630		$6n-2$ 2,1
631		$7n-3$ 2,1*
632		$6n-2$ 2,1
633		$6n-4$ 2,1
634		$8n-4$ 3,1*
635		$8n-2$ 2,1
636		$9n-3$ 2,1*
637		$8n-2$ 2,1*
638		$8n-2$ 2,1*
639		$8n-4$ 4,2
640		$8n-2$ 2,1*
641		$10n-2$ 2,1*
642		$7n-3$ 2,1

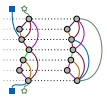
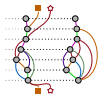
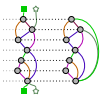
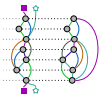
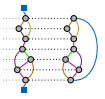
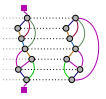
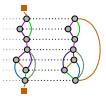
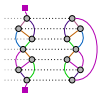
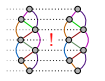
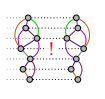
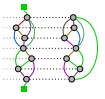
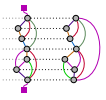
i	G_i^7	cr d,m
643		$9n-3$ $2,1^*_*$
644		$9n-5$ 4,2
645		$9n-3$ $2,1^*_*$
646		$5n-1$ 2,1
647		$7n-1$ 2,1
648		$7n-1$ 2,1
649		$7n-3$ 4,2
650		$8n+0$ 4,2
651		$9n-3$ $2,1^*_*$
652		$9n-1$ 2,1
653		$11n-1$ $2,1^*_*$
654		$9n-1$ $2,1^*_*$
655		$9n-1$ $2,1^*_*$

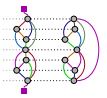
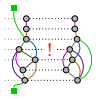
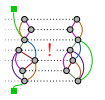
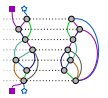
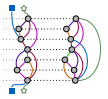
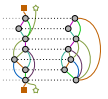
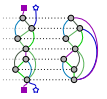
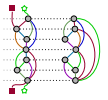
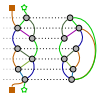
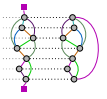
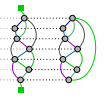
i	G_i^7	cr d,m
657		$10n-2$ $2,1^*_*$
658		$10n-2$ $4,2^*_*$
659		$12n-2$ $3,1^*_*$
660		$7n-1$ $2,1^*_*$
661		$7n-3$ 4,2
662		$7n-3$ 2,1
664		$8n-2$ 3,1
665		$8n-2$ $3,1^*$
666		$8n-2$ $3,1^*_*$
667		$8n-2$ $2,1^*_*$
668		$10n-2$ $3,1^*_*$
670		$9n-3$ 3,1
672		$9n-1$ 2,1

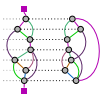
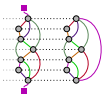
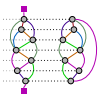
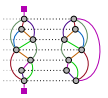
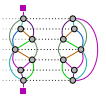
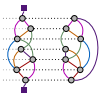
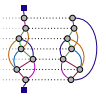
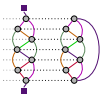
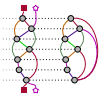
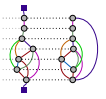
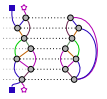
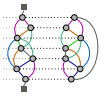
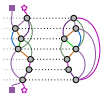
i	G_i^7	cr d,m
673		$10n-2$ 2,1
676		$11n-3$ $2,1^*_*$
677		$12n-2$ $2,1^*_*$
680		$11n-3$ 3,1
682		$9n-1$ $2,1^*_*$
683		$12n-2$ $2,1^*_*$
687		$11n-1$ $2,1^*_*$
688		$10n-4$ $2,1^*_*$
690		$12n-2$ $2,1^*_*$
692		$8n-2$ 2,1
693		$7n-1$ 2,1
694		$8n+0$ $3,1^*_*$
695		$6n+0$ 2,1

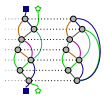
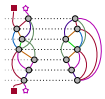
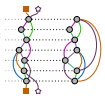
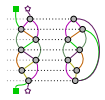
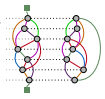
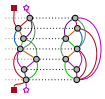
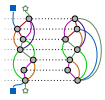
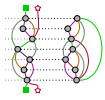
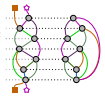
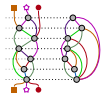
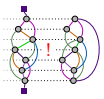
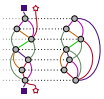
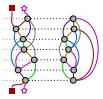
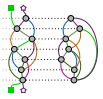
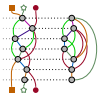
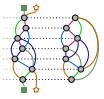
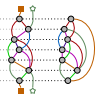
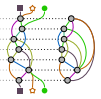
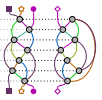
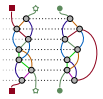
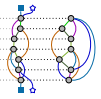
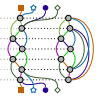
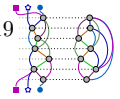
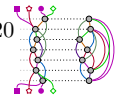
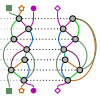
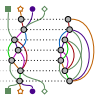
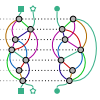
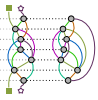
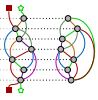
i	G_i^7	cr d,m
696		$9n-3$ $3,1^*_*$
697		$6n+0$ 2,1
698		$10n-2$ $3,1^*_*$
699		$9n-3$ 2,1
700		$9n-3$ 2,1
701		$8n-2$ 2,1
702		$11n-3$ $2,1^*_*$
703		$10n-2$ 2,1
704		$11n-1$ $2,1^*_*$
705		$10n-2$ $2,1^*_*$
706		$12n-2$ $2,1^*_*$
707		$9n-1$ 2,1
708		$9n-1$ $2,1^*_*$

i	G_i^7	cr d,m
709		$11n-1$ 2,1*
711		$13n-1$ 2,1
712		$12n+0$ 2,1
714		$12n-2$ 2,1
718		$10n-2$ 2,1
728		$3n-3$ 3,1
729		$4n-4$ 2,1
730		$4n-2$ 3,1*
731		$5n-3$ 2,1*
732		$6n-2$ 2,1*
733		$5n-1$ 2,1
735		$7n-5$ 3,1*
736		$6n-2$ 2,1

i	G_i^7	cr d,m
737		$8n-6$ 3,1*
738		$8n-4$ 4,2*
739		$6n-4$ 2,1
740		$7n-3$ 4,2
741		$8n-4$ 3,1*
742		$3n-1$ 2,1*
743		$5n-3$ 2,1*
744		$5n-3$ 2,1*
745		$4n+0$ 2,1
746		$5n-3$ 3,1*
747		$6n-4$ 4,2*
748		$6n-2$ 4,2
749		$7n-5$ 3,1*

i	G_i^7	cr d,m
750		$8n-4$ 4,2
751		$6n-2$ 3,1
752		$7n-3$ 4,2*
755		$7n-1$ 2,1
756		$9n-5$ 4,2*
757		$8n-4$ 4,2
758		$7n-3$ 2,1*
759		$9n-3$ 2,1*
764		$10n-2$ 2,1*
765		$12n-4$ 2,1*
766		$6n+0$ 2,1
767		$6n-2$ 2,1
768		$6n-2$ 2,1

i	G_i^7	cr d,m
769		$7n-1$ 2,1
770		$7n-5$ 3,1*
771		$7n-1$ 2,1
772		$8n-4$ 2,1*
773		$9n-3$ 2,1*
774		$8n-2$ 2,1
775		$7n-1$ 2,1
776		$5n-1$ 2,1
777		$6n-2$ 2,1
778		$8n-2$ 3,1*
779		$5n-1$ 2,1
780		$8n-2$ 2,1
781		$9n-3$ 3,1

i	G_i^7	cr d,m
782		$7n-3$ 2,1
783		$10n-4$ $2,1^*_*$
784		$8n-2$ 3,1
785		$6n+0$ 2,1
786		$9n-1$ 2,1
787		$10n+0$ $4,2^*_*$
788		$11n-3$ $2,1^*_*$
789		$10n-4$ $3,1^*_*$
791		$10n-4$ $2,1^*_*$
794		$8n-2$ 2,1
795		$10n-2$ $2,1^*_*$
796		$10n-2$ $2,1^*_*$
797		$9n-3$ 2,1
798		$12n-2$ $2,1^*_*$
799		$13n-1$ $3,1^*_*$
800		$12n+0$ 2,1
801		$11n-1$ 2,1
802		$14n-2$ $2,1^*$
804		$8n-2$ 2,1
805		$8n-2$ 2,1
806		$11n-3$ $3,1^*_*$
807		$9n-3$ 2,1
808		$10n-2$ 2,1
809		$11n-3$ $2,1^*_*$
810		$10n+0$ 2,1
811		$13n-1$ $2,1^*_*$
812		$8n+0$ 2,1
813		$9n-1$ 2,1
814		$7n-1$ 2,1
815		$10n-2$ 2,1
817		$8n+0$ 2,1
818		$10n-2$ 2,1
819		$11n-3$ 2,1
820		$13n-3$ $2,1^*_*$
821		$10n+0$ 2,1
822		$12n-4$ $3,1^*_*$
823		$13n-1$ $2,1^*_*$
825		$13n-1$ 2,1
826		$12n+0$ 2,1
827		$15n-1$ 2,1
830		$13n-1$ 2,1
831		$11n-1$ $2,1^*_*$
833		$13n+1$ $2,1^*_*$

On Geometric Bipartite Graphs with Asymptotically Smallest Zarankiewicz Numbers

Parinya Chalermsook
University of Sheffield, England

Ly Orgo
Aalto University, Finland

Minoo Zarsav
Aalto University, Finland

Abstract

This paper considers the *Zarankiewicz problem* in bipartite graphs with low-dimensional geometric representation (i.e., low Ferrers dimension). [1] Let $Z(n;k)$ be the maximum number of edges in a bipartite graph with n nodes and is free of a k-by-k biclique. Note that $Z(n;k) \in \Omega(nk)$ for all "natural" graph classes. Our first result reveals a separation between bipartite graphs of Ferrers dimension three and four: while we show that $Z(n;k) \leq 9n(k-1)$ for graphs of Ferrers dimension three, $Z(n;k) \in \Omega\left(nk \cdot \frac{\log n}{\log\log n}\right)$ for Ferrers dimension four graphs (Chan & Har-Peled, 2023) (Chazelle, 1990). To complement this, we derive a tight upper bound of $2n(k-1)$ for chordal bipartite graphs and $54n(k-1)$ for grid intersection graphs (GIG), a prominent graph class residing in four Ferrers dimensions and capturing planar bipartite graphs as well as bipartite intersection graphs of rectangles. Previously, the best-known bound for GIG was $Z(n;k) \in O(2^{O(k)}n)$, implied by the results of Fox & Pach (2006) and Mustafa & Pach (2016). Our results advance and offer new insights into the interplay between Ferrers dimensions and extremal combinatorics.

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

Keywords and phrases Bipartite graph classes, extremal graph theory, geometric intersection graphs, Zarankiewicz problem, bicliques

Digital Object Identifier 10.4230/LIPIcs.GD.2025.21

Funding This work was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 759557).

1 Introduction

Bipartite graphs are the most natural mathematical objects used to capture interrelations between two groups of interest. Therefore, it is unsurprising that they arise routinely in mathematical and algorithmic research communities. In extremal graph theory, bipartite graphs have been studied extensively from several perspectives.

The Zarankiewicz problem is perhaps among the oldest questions in extremal graph theory. In 1951, Zarankiewicz asked for the maximum number of edges in a bipartite graph with n nodes on each side that does not contain a complete bipartite subgraph $K_{k,k}$. [2] Equivalently, given an n-by-n matrix with 0/1 entries, that does not contain a k-by-k all-one submatrix, what is the maximum number of 1-entries in such a matrix? This question was partially answered by Kövari, Sós and Turán: If we denote such number by $Z(n;k)$, it was proven that $Z(n;k) \in O(n^{2-1/k})$ [20], and the best known lower bound is $\widetilde{\Omega}(n^{2-2/(k+1)})$

[1] Ferrers dimension, along with interval dimension and order dimension, is a standard dimensional concept in graphs. Please refer to the introduction for formal definition.

[2] Here, we are interested in a symmetric form of the Zarankiewicz problem. A more general question can involve graphs with different numbers of vertices on each side, that do not contain some biclique $K_{s,t}$.



© Parinya Chalermsook, Ly Orgo, and Minoo Zarsav;
licensed under Creative Commons License CC-BY 4.0



shown in [2]. Exact values are known for $k = 2$, $k = 3$, and closing the gap for $k \geq 4$ is one of the central open questions in extremal combinatorics. A trivial lower bound would be $n \cdot k$, achieved by $K_{k,n}$.

This question has also attracted significant interest in the context of specific graph classes, particularly those with additional structural constraints [4, 25, 12]. Chan and Har-Peled, for instance, studied a wide range of geometrically defined bipartite graphs [4], such as incidence graphs of points and rectangles, disks, or pseudodisks.

1.1 Our contributions

We consider the Zarankiewicz problem for geometric bipartite graphs. For "natural" graph classes, the lower bound of $Z(n;k) \in \Omega(nk)$ holds trivially. Our results show that a nearly tight upper bound can be achieved for many natural graph classes.

To formalize the discussion, consider a bipartite intersection graph $G = (U \cup V, E)$, defined by two families of objects $\mathcal{F}_U$ and $\mathcal{F}_V$, where each vertex $u \in U$ (resp. $v \in V$) is associated with an object $O_u \in \mathcal{F}_U$ (resp. $O_v \in \mathcal{F}_V$); Let $\phi : u \mapsto O_u, v \mapsto O_v$ for $u \in U$ and $v \in V$ be the bijection between vertices and their representations. We say that $(\mathcal{F}_U, \mathcal{F}_V, \phi)$ is an **intersection representation** of G. Whenever the mapping ϕ is clear from the context, we omit it. Table 1 shows bipartite intersection graphs relevant to this paper.

Table 1 Classes of bipartite intersection graphs that are relevant to this paper.

Graph Classes	**Family** $\mathcal{F}_U$	**Family** $\mathcal{F}_V$	domain	**Example**
CHAIN	rightward rays	points	$\mathbb{R}$	
CONV	intervals	points	$\mathbb{R}$	
CHAIN^2	rightward rays	upward rays	$\mathbb{R}^2$	
SR	horizontal segments	upward rays	$\mathbb{R}^2$	
GIG	horizontal segments	vertical segments	$\mathbb{R}^2$	
PRIG	axis-aligned rectangles	points	$\mathbb{R}^2$	

For two graph classes $\mathcal{C}_1$ and $\mathcal{C}_2$, we use $\mathcal{C}_1 \cdot \mathcal{C}_2$ to denote the class of graphs, such that every graph $G \in \mathcal{C}_1 \cdot \mathcal{C}_2$ is the intersection of some graphs $G_1 \in \mathcal{C}_1$, $G_2 \in \mathcal{C}_2$, where intersection between graphs is defined as the intersection between their nodes and edges. When $\mathcal{C}_1 = \mathcal{C}_2$, we use the notation $\mathcal{C}_1^2 = \mathcal{C}_1 \cdot \mathcal{C}_1$, and similarly for other powers.

The concept of Ferrers dimension is crucially based on **chain graphs** (CHAIN). A bipartite graph $G = (U \cup V, E)$ is in CHAIN if it is an intersection graph of rays and points on the line [7]. Ferrers dimension of a bipartite graph G, denoted by $\mathsf{fd}(\mathsf{G})$, is the smallest integer d, such that $G \in \mathsf{CHAIN}^d$. It is known that every n-vertex graph has Ferrers dimension at most n.

We will use $Z_{\mathcal{C}}(n, m; k, k)$ to denote the maximum number of edges in a bipartite graph $G = (U \cup V, E) \in \mathcal{C}$, where $|U| = n$, $|V| = m$, such that the graph does not contain $K_{k,k}$ as a subgraph, and simply $Z_{\mathcal{C}}(n; k)$, when $n = m$.

Our first main result provides a dichotomy between Ferrers dimension three and four.

▶ **Theorem 1** (Dichotomy theorem). *The following dichotomy results hold for all $k \in \mathbb{N}$:*

- $Z_{\mathsf{CHAIN}^3}(n; k) \leq 9n(k-1)$.
- $Z_{\mathsf{CHAIN}^4}(n; k) \in \Omega(nk \cdot \frac{\log n}{\log \log n})$.

By using our upper bound for Ferrers dimension three as a base case, a result by Chan and Har-Peled [4] can be improved (proof in Appendix A).

▶ **Corollary 2.** *For graphs of Ferrers dimension $d \geq 3$, $k \in \mathbb{N}$, we have $Z_{\mathsf{CHAIN}^d}(n;k) \leq O(nk \cdot \lceil \log n \rceil^{d-3})$.*

We complement the lower bound for CHAIN^4 by providing Zarankiewicz upper bounds for a prominent graph class in CHAIN^4.

▶ **Theorem 3.** *For all $k, n \in \mathbb{N}$, $Z_{\mathsf{GIG}}(n;k) \leq 54n(k-1)$.*

This is our second main result. Grid intersection graph (GIG) is an intersection graph of horizontal and vertical segments in $\mathbb{R}^2$. This graph class contains all graphs of Ferrers dimension two and is known to be in CHAIN^4. GIG is rich, includes all planar bipartite graphs, and is equivalent to the bipartite intersection graphs of rectangles [1]. They have been studied from both structural and algorithmic perspectives [8, 22, 21]. This result improves upon the $2^{O(k)}n$ upper bound that follows from [25, 13] (their results hold for more general intersection graphs of curves).[3]

Our final result applies to chordal bipartite graphs – a bipartite graph is chordal if every cycle of length at least six contains a chord. Chordal bipartite graphs contain all graphs of Ferrers dimension at most two. On the other hand, it is relatively rich, having unbounded Ferrers dimension [7, 6].

▶ **Theorem 4.** *For all $k, n \in \mathbb{N}$, $Z_{\mathcal{C}}(n;k) \leq 2n(k-1)$, where $\mathcal{C}$ is a class of chordal bipartite graphs.*

We observe that $Z_{\mathsf{CHAIN}}(n;k) \geq 2n(k-1) - O(k^2)$ even for CHAIN (see Lemma 38 in Appendix E), so our upper bound is tight up to an additive constant for fixed k. Theorem 4 improves upon the $4nk$ upper bound that holds for CONV [4] (since such graphs have Ferrers dimension at most two). Please refer to Table 2 for a comprehensive list of our results and Figure 1 for a landscape of $Z_{\mathcal{C}}(n;k)$, when $\mathcal{C}$ is one of these graph chasses.

Table 2 Zarankiewicz Bounds proved in this paper. Our bounds follow from Theorem 4, Corollary 13, Corollary 19, and Theorem 3 respectively.

Graph classes	**Our UB**	**Known UB**
Chordal bipartite graphs	$2n(k-1)$	$3kn$ for a special case. [4]
Segment ray graphs (SR)	$4n(k-1)$	$O(kn)$ [4]
CHAIN^3	$9n(k-1)$	
Grid intersection graphs (GIG)	$54n(k-1)$	$2^{O(k)}n$ [13, 25]

We remark that all our upper bounds are algorithmic in the sense that, given a graph and its geometric representation in class $\mathcal{C}$, we give an efficient algorithm that reports a k-by-k biclique whenever the number of edges exceeds our $Z_{\mathcal{C}}(n;k)$ upper bound (e.g., given a grid intersection graph $\mathcal{C} = \mathsf{GIG}$ with more than $54(k-1)n$ edges, our algorithm efficiently finds a k-by-k biclique).

[3] Note that Fox and Pach explicitly mentioned the term $2^{O(k)}$, while Mustafa and Pach's dependency on k is somewhat hidden in their calculation. Looking at their calculations reveals that the constant is $2^{O(k)}$.

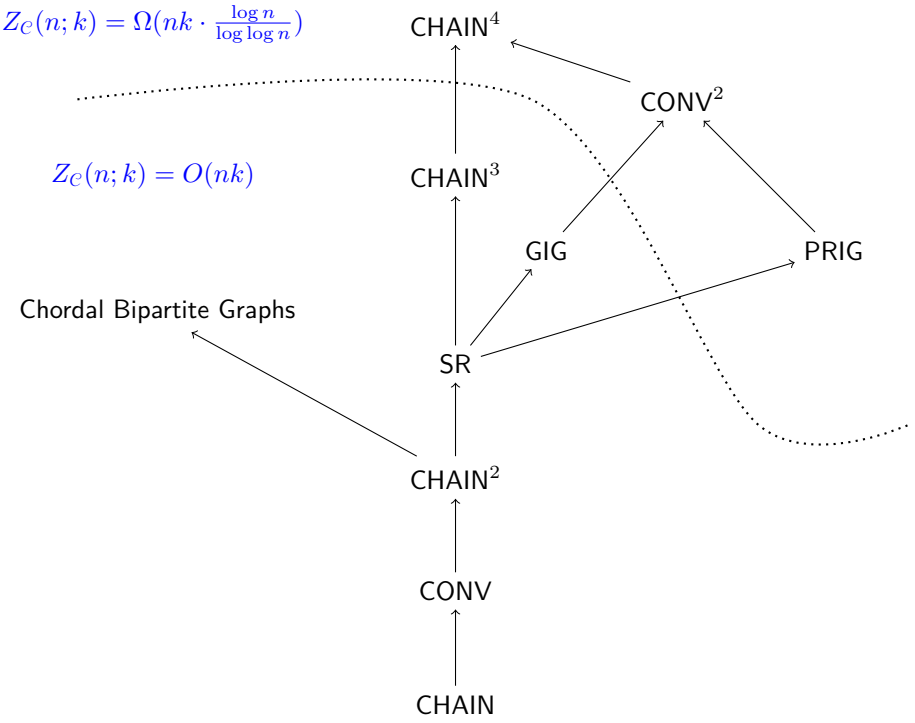


Figure 1 The relation between graph classes considered in this paper. All inclusions denoted by arrows are known to be proper. Our results imply the separation shown by the dotted curve. We show that CHAIN^3, GIG, and chordal bipartite graphs satisfy $Z_{\mathcal{C}}(n;k) \in O(nk)$, which implies the same for the rest of the graph classes below them.

Comparison to known results. Prior to our results, the upper bound $Z_{\mathcal{C}}(n;k) \in O(nk)$ has been known for CONV, segments and rays[4], and halfspaces and points [4]. We improve a constant for CONV (since it is a special case of chordal bipartite graphs) and generalize their result on the segment-ray graphs to grid intersection graphs. The grid intersection graph is a special case of string graphs, so the upper bound [13, 25] of $Z_{\mathsf{GIG}}(n;k) \in O(2^{O(k)}n)$ follows from their results on string graphs. Please refer to Figure 1 for the relations between these classes.

Further related works. The graph classes in this paper have received much attention from various perspectives, including recognition algorithms, optimization, and structures. For instance, chordal bipartite graphs have been considered in [23, 15, 24, 30]. For grid intersection graphs, the recognition problem is NP-complete [21, 22]. The class is known to capture planar bipartite graphs [16] and is equivalent to the intersection graph of rectangles that are bipartite [1]. For more detailed literature on GIG, we refer the readers to an exposition in the PhD thesis of Mustata [26].

Zarankiewicz problems have been studied in many other special graphs, such as semi-algebraic graphs [14, 10], graphs of bounded VC dimension [17], graphs forbidding a fixed induced subgraph [3], and in geometric intersection graphs [18, 5, 4, 29, 5]. We refer to the recent survey by Smorodinsky for a more comprehensive list of related works [28].

[4] Chan and Har-Peled use the equivalent 3-sided rectangle and point intersection graphs.

2 Preliminaries

We use standard graph-theoretic notation. A bipartite graph is denoted as $G = (U \cup V, E)$ where U and V are partitions of the vertices. For a graph G, we refer to its set of vertices as $V(G)$, its set of edges as $E(G)$, and its induced subgraph on $S \subseteq V(G)$ as $G[S]$.

The intersection of two graphs G_1 and G_2 is defined as $G_1 \cap G_2 = (V(G_1) \cap V(G_2), E(G_1) \cap E(G_2))$. When G_1 belongs in graph class $\mathcal{C}_1$ and G_2 in class $\mathcal{C}_2$, then we say that $G_1 \cap G_2$ belongs in $\mathcal{C}_1 \cdot \mathcal{C}_2$. For graph classes $\mathcal{C}_1$ and $\mathcal{C}_2$, that are closed under taking induced subgraphs, a graph $G \in \mathcal{C}_1 \cdot \mathcal{C}_2$ can be assumed to be the intersection of two graphs on the same vertex set, because if $G_1 \in \mathcal{C}_1$, then $G_1[V(G_2) \cap V(G_1)] \in \mathcal{C}_1$. Similarly for G_2.

Therefore, for two graph classes $\mathcal{C}_1$ and $\mathcal{C}_2$ that are closed under taking induced subgraphs, the intersection of the graph classes can be defined as:

$$\mathcal{C}_1 \cdot \mathcal{C}_2 = \{G_1 \cap G_2 : V(G_1) = V(G_2), G_1 \in \mathcal{C}_1, G_2 \in \mathcal{C}_2\}$$

One can naturally write $\mathcal{C}^d = \mathcal{C} \cdot \mathcal{C} \dots \mathcal{C}$ (d times).

Bicliques are the graphs $K_{a,b}$, where a and b are some positive integers.

▶ **Proposition 5.** *If the graph class $\mathcal{C}$ contains every biclique, then $\mathcal{C} \subseteq \mathcal{C}^2 \subseteq \dots$.*

Proof. Here, we show that for any positive integer i, $\mathcal{C}^i \subseteq \mathcal{C}^{i+1}$. Let $G = (U \cup V, E) \in \mathcal{C}^i$. Then take H being the complete bipartite graph on U and V (where the edges connect vertices in U and V). Then $H \in \mathcal{C}$. Therefore, $G = G \cap H \in \mathcal{C}^{i+1}$. ◀

Since CHAIN contains every biclique, this implies that $\mathsf{CHAIN} \subseteq \mathsf{CHAIN}^2 \subseteq \dots$. It is known that CHAIN^2 is equivalent to the class of all two-directional orthogonal ray graphs (2DOR) [7, Theorem 2.1].

A bipartite graph $G = (U \cup V, E)$ is **convex** over V if there exists an ordering $V = \{v_1, \dots v_{|V|}\}$ such that for all $u \in U$, the vertices $N(u)$ are contiguous ($\{v_i, \dots v_j\}$ for some $i, j \in [|V|]$). We call the set of such graphs CONV. CONV is known to be equivalent to a containment graph of points and intervals in $\mathbb{R}$. An interesting graph class worth mentioning is CONV^2 – the intersection of two convex graphs.

▶ **Theorem 6.** *A bipartite graph $G = (U \cup V, E)$ is in CONV^2 if and only if G is a disjoint union of GIG and PRIG graphs.*

The proof can be found in Appendix B.

Finally, a bipartite graph is a **chordal bipartite graph** if every induced cycle of length at least six contains a chord. It is known that chordal bipartite graphs can have unbounded Ferrers dimension [7, 6].

3 Warm-up

Recall that a graph G is d-**degenerate** if every induced subgraph has a vertex of degree at most d. All graph classes considered in this paper are closed under taking induced subgraphs.

▶ **Observation 7.** *A d-degenerate graph G has at most $d \cdot |V(G)|$ edges.*

3.1 Chordal bipartite graphs

For a bipartite graph G, denote its biadjacency matrix by M_G. We say that a matrix M **contains** submatrix P if we can obtain P by removing some rows and columns of M. A matrix M is P-**free** if it does not contain submatrix P. A bipartite graph G is P-**freeable**,

if there exists an ordering of rows and columns, such that the biadjacency matrix M_G is P-free.

The following structural result is known for chordal bipartite graphs.

▶ **Theorem 8** ([19, 11]). *Every chordal bipartite graph is* $\left(\begin{smallmatrix}0&1\\1&1\end{smallmatrix}\right)$*-freeable.*

▶ **Lemma 9.** *For any* $k \in \mathbb{N}$*, a chordal bipartite graph that does not contain* $K_{k,k}$ *is* $(k-1)$*-degenerate.*

Proof. Let H be a chordal bipartite graph. If for every induced subgraph of H, a node of degree at most $k-1$ exists, we are done. Therefore, let us assume by contradiction that G is an induced subgraph of H, where a node of degree at most $k-1$ does not exist. Then all nodes $V(G)$ have a degree of at least k in G. Using Theorem 8, we know that there is a P-free biadjacency matrix of G, where $P = \left(\begin{smallmatrix}0&1\\1&1\end{smallmatrix}\right)$. Let M_G be that biadjacency matrix. Let $G = (U \cup V, E)$, where each row $i \in [|U|]$ and column $j \in [|V|]$ correspond to the vertices $u_i \in U$ and $v_j \in V$, respectively. Let $i \in [|U|]$ be the maximum integer, such that $M_G(i, |V|) = 1$ (i.e., this is the bottommost row in the last column such that the entry is one). Consider the rows $R = \{i' : u_{i'} \in N_G(v_{|V|})\} \subseteq [|U|]$ and columns $C = \{j' : v_{j'} \in N_G(u_i)\} \subseteq [|V|]$. Let M' be the submatrix induced on rows R and columns C.

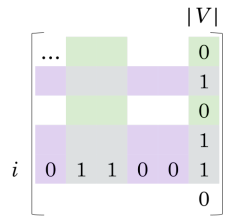


Figure 2 An example, where rows R are denoted in purple and columns C in green.

According to our assumption, this submatrix M' contains at least k rows and k columns. It is easy to see that M' is an all-one matrix: Assume that it is not, and $M_G(a, b) = 0$ for some $a \in R$ and $b \in C$. Since $a \in R$, we have $M_G(a, |V|) = 1$ and since $b \in C$, $M_G(i, b) = 1$. The 2-by-2 submatrix induced on rows $\{a, i\}$ and columns $\{b, |V|\}$ is then equal to the forbidden pattern P, a contradiction. ◀

▶ **Theorem 10.** *For all* $n, m, k \in \mathbb{N}$*,* $Z_{\mathcal{C}}(m, n; k, k) \leq (m+n)(k-1)$*, where* $\mathcal{C}$ *is the class of chordal bipartite graphs.*

Proof. The result follows from Lemma 9 and Observation 7. ◀

The matching lower bound is shown in Lemma 38.

▶ **Theorem 4.** *For all* $k, n \in \mathbb{N}$*,* $Z_{\mathcal{C}}(n; k) \leq 2n(k-1)$*, where* $\mathcal{C}$ *is a class of chordal bipartite graphs.*

Proof. The result follows from Theorem 10 when $n = m$. ◀

3.2 Segment ray graphs

Next, we present a simple proof of the upper bound of $Z_{\mathsf{SR}}(m, n; k, k)$ for the intersection graph of horizontal segments and vertical upward rays in $\mathbb{R}^2$, denoted by SR. More precisely, $(\mathcal{S}, \mathcal{R}, \phi)$, is a segment ray ($\mathsf{SR}$) representation of a bipartite graph $G = (U \cup V, E)$, if $\mathcal{S}$

and $\mathcal{R}$ are sets of horizontal segments and vertical upwards rays in $\mathbb{R}^2$ respectively, and $\phi : U \mapsto \mathcal{S}, V \mapsto \mathcal{R}$, such that $\{u, v\} \in E$, where $u \in U$, $v \in V$, if and only if $\phi(u)$ and $\phi(v)$ intersect.

Although the upper bound $O((m+n)k)$ was shown in [4], we improve this upper bound to $2(m+n)(k-1)$.

▶ **Lemma 11.** *Every segment-ray graph that does not contain $K_{k,k}$ is $2(k-1)$-degenerate.*

Proof. Let H be a segment-ray graph. If, for every induced subgraph of H, a node of degree at most $2(k-1)$ exists, we are done. Therefore, let us assume by contradiction, that G is an induced subgraph of H, where a node of degree at most $2(k-1)$ does not exist. Then, all nodes $V(G)$ must have a degree at least $2k-1$ in G.

Let $(\mathcal{S}, \mathcal{R}, \phi)$ be the SR representation of $G = (U \cup V, E)$, such that $\phi : U \mapsto \mathcal{S}, V \mapsto \mathcal{R}$. Let x and y be the horizontal and vertical axes, respectively. Each ray $r \in \mathcal{R}$ can be addressed by a pair (r_x, r_y), where r_x and r_y are the $\mathbf{x}$ and $\mathbf{y}$ coordinates of its starting point respectively. Each horizontal segment $s \in \mathcal{S}$ can be addressed using three variables (s_l, s_r, s_y), such that $[s_l, s_r]$ and s_y are the projections of s to the x and y axis respectively.

Let $r \in \mathcal{R}$ be the ray with the largest r_y (highest starting point). Let S be the set of segments that intersect r. Then, according to our assumption, $|S| \geq 2k-1$. Let $S_{right} \subseteq S$ be the set of $k-1$ segments with the largest (rightmost) s_r, and let $S_{left} \subset S$ be the set of $k-1$ segments with the least (leftmost) s_l. Then there must be at least one segment $s \in S \setminus (S_{right} \cup S_{left})$ (Figure 3).

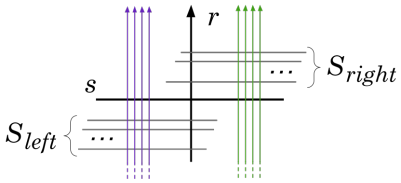


Figure 3 R_{left} and R_{right} are denoted by purple and green rays respectively.

Let R be the set of rays that intersect s. By our assumption, $|R| \geq 2k-1$. Let $R_{left}, R_{right} \subseteq R$ be the sets of rays to the left and right of r, respectively. Since $R = R_{left} \cup R_{right} \cup \{r\}$, then at least one of R_{left}, R_{right} must contain at least $k-1$ rays. Let us assume it is R_{left}, the other case can be argued symmetrically. Since r is the ray with the highest r_y, then each ray in R_{left} must intersect each segment in S_{left} (Figure 3). This implies that the vertices mapped to rays in $R_{left} \cup \{r\}$ and segments in $S_{left} \cup \{s\}$ must induce a k-by-k biclique. This is a contradiction. ◀

▶ **Theorem 12.** *For all $n, m, k \in \mathbb{N}$, $Z_{\mathsf{SR}}(m, n; k, k) \leq 2(m+n)(k-1)$ in segment ray graphs.*

Proof. The result follows from Lemma 11 and Observation 7. ◀

▶ **Corollary 13.** *When $m = n$, $Z_{\mathsf{SR}}(n; k) \leq 4n(k-1)$ in segment ray graphs.*

4 Graphs of Ferrers Dimension Three

This section presents the proof of the first part of Theorem 1.

Let x and y be the horizontal and vertical axes, respectively. We say that a rectangle r in $\mathbb{R}^2$ is bottomless [5] if its projection to the y-axis is an interval that starts at $-\infty$.

We say that $(\mathcal{S}, \mathcal{B}, \phi)$ is a segment bottomless rectangle containment representation of a bipartite graph $G = (U \cup V, E)$, if $\mathcal{S}$ and $\mathcal{B}$ are sets of horizontal segments and bottomless rectangles in $\mathbb{R}^2$, respectively, $\phi : U \mapsto \mathcal{S}, V \mapsto \mathcal{B}$, and for all $u \in U$, $v \in V$, we have $\{u, v\} \in E$ if and only if $\phi(u)$ is contained within $\phi(v)$. A bipartite graph G is a segment bottomless rectangle containment graph, if it has a segment bottomless rectangle containment representation.

For $s \in \mathcal{S} \cup \mathcal{B}$, let $s.x$ and $s.y$ denote the projections of s to x- and y-axis respectively. For an interval $s.x = [a, b]$, we use $\min(s.x) = a$ and $\max(s.x) = b$ to denote its minimum and maximum points.

▶ **Lemma 14.** *A bipartite graph G is of Ferrers dimension three ($G \in \mathsf{CHAIN}^3$) if and only if it is a segment bottomless rectangle containment graph.*

The proof can be found in Appendix C.

Given a bipartite graph $G \in \mathsf{CHAIN}^3$, we now know it must have a containment representation $(\mathcal{S}, \mathcal{B}, \phi)$, where $\mathcal{S}$ and $\mathcal{B}$ are sets of horizontal segments and bottomless rectangles in $\mathbb{R}^2$, and $\phi : U \mapsto \mathcal{S}, V \mapsto \mathcal{B}$.

Let k be the minimum positive integer such that G does not contain $K_{k,k}$ as a subgraph.

Without loss of generality, assume that no two horizontal segments have the same y-coordinate. We define three (strict) partial orders (denoted by $\mathcal{P}^{DL}$, $\mathcal{P}^{DR}$, and $\mathcal{P}^{C}$) on $\mathcal{S}$ that will be crucial to our analysis (Figure 4). For $s, s' \in \mathcal{S}$, we say

- s' succeeds s in the partial order $\mathcal{P}^{DL}$, and denote it by $s \prec_{DL} s'$, if $\min(s'.x) \leq \min(s.x)$, $\max(s'.x) \leq \max(s.x)$, and $s'.y < s.y$,
- s' succeeds s in the partial order $\mathcal{P}^{DR}$, and denote it by $s \prec_{DR} s'$, if $\min(s.x) \leq \min(s'.x)$, and $\max(s.x) \leq \max(s'.x)$, and $s'.y < s.y$,
- s' succeeds s in the partial order $\mathcal{P}^{C}$, and denote it by $s \prec_{C} s'$, if $s'.x \subset s.x$.

It is easy to check that $\mathcal{P}^{DL}$, $\mathcal{P}^{DR}$, and $\mathcal{P}^{C}$ are transitive and asymmetric.

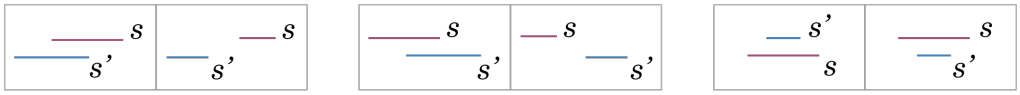


Figure 4 The scenarios that define relations in $\mathcal{P}^{DL}$ (two leftmost), $\mathcal{P}^{DR}$ (two in the middle) and $\mathcal{P}^{C}$ (two rightmost images), where s' succeeds s.

If $s \prec_{DL} s'$ or $s' \prec_{DL} s$, we say that s and s' are comparable in $\mathcal{P}^{DL}$; otherwise, they are incomparable. We can say the same about comparability in $\mathcal{P}^{DR}$ and $\mathcal{P}^{C}$. The following claim is straightforward, and a consequence of a simple case analysis (see Figure 4).

▶ **Observation 15.** *A pair of segments $s, s' \in \mathcal{S}$ is comparable in one of the partial orders $\mathcal{P}^{DL}, \mathcal{P}^{DR}, \mathcal{P}^{C}$.*

For a segment $s \in \mathcal{S}$, denote by $\mathsf{succ}^{DL}(s), \mathsf{succ}^{DR}(s), \mathsf{succ}^{C}(s) \subseteq \mathcal{S}$ the sets of all successors of s with respect to partial orders $\mathcal{P}^{DL}$, $\mathcal{P}^{DR}$ and $\mathcal{P}^{C}$, respectively.

To count the number of edges in a bipartite graph G with a representation $(\mathcal{S}, \mathcal{B}, \phi)$, we classify the edges into **bulky** and **thin** edges. For an edge $\{u, v\} \in E(G)$, we have containment $\phi(u) \subseteq \phi(v)$, where $\phi(u) \in \mathcal{S}$, $\phi(v) \in \mathcal{B}$. We say that the edge $\{u, v\}$ is

[5] These rectangles have also been called 3-sided rectangles [4] in the context of SR graphs.

DL-bulky if the bottomless rectangle $\phi(v)$ contains $k-1$ segments from $\mathsf{succ}^{DL}(\phi(u))$ (See Figure 5). We can analogously define **DR-bulky** and **C-bulky** edges. All edges that are not bulky are called thin. An edge is bulky if it is DL-bulky, DR-bulky, or C-bulky.

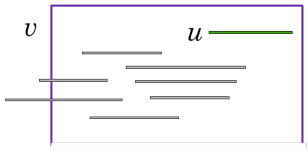


Figure 5 The edge $\{u, v\}$ is DL-bulky, if $\phi(v)$ (purple) contains $k-1$ segments succeeding $\phi(u)$ (green) in $\mathscr{P}^{DL}$.

If $s \prec_{DL} s'$ or $s' \prec_{DL} s$, we say that s and s' are comparable in $\mathscr{P}^{DL}$; otherwise, they are incomparable. We can say the same about comparability in $\mathscr{P}^{DR}$ and $\mathscr{P}^{C}$. The following claim is straightforward, and a consequence of a simple case analysis (see Figure 4).

▶ **Lemma 16.** *For each $u \in U$, the number of bulky edges incident to u of each type is at most $(k-1)$. Therefore, at most $3(k-1)$ bulky edges are incident to each u.*

Proof. We present the proof for DL-bulky edges. The proofs for the other two cases are similar. Let $B \subseteq N_G(u)$ be the nodes $v \in V$, such that $\{u, v\}$ is DL-bulky. Let $S' \subseteq \mathsf{succ}^{DL}(\phi(u))$ be the $k-1$ segments $s' \in S'$ with the largest $\min(s'.x)$. Let U' be the set of nodes represented by S'.

We will show that each $v \in B$ has $\{u\} \cup U'$ in its neighborhood, and therefore $|B| \leq k-1$. Assume by contraposition that $|B| \geq k$. Let $v \in B$ be an arbitrary node. For any $s' \in S'$, $\phi(u) \prec_{DL} s'$ and $\phi(u) \subseteq \phi(v)$ imply the following:

- $s'.y \in \phi(v).y$. Since $\phi(u) \prec_{DL} s'$, we have $s'.y \leq \phi(u).y$, and since $\phi(v)$ contains $\phi(u)$, we have $\phi(u).y \in \phi(v).y$. Since $\phi(v).y$ is an interval that starts at $-\infty$, then $s'.y \in \phi(v).y$.
- $\max(s'.x) \leq \max(\phi(v).x)$. Similarly to the previous case, the implications give us the inequalities in $\max(s'.x) \leq \max(\phi(u).x) \leq \max(\phi(v).x)$

Let $s' \in S'$ be an arbitrary segment. Since $\phi(v).y$ contains s' on the y-axis, $\max(\phi(v).x)$ is larger than any point in s' on the x-axis, then to show that $\phi(v)$ contains s', it only remains to show that $\min(\phi(v).x) \leq \min(s'.x)$. Since S' was chosen to contain the segments in $\mathsf{succ}^{DL}(\phi(u))$ with the largest $\min(s'.x)$, then $\min(\phi(v).x) \leq \min(s'.x)$ holds for $s' \in S'$, if it holds for any $k-1$ segments in $\mathsf{succ}^{DL}(\phi(u))$. Since $\{u, v\}$ is DL-bulky, $\phi(v)$ contains at least $k-1$ segments in $\mathsf{succ}^{DL}(\phi(u))$. This implies that $\phi(v)$ contains S', and we have $U' \in N_G(v)$. Since this is true for all $v \in B$, then B and $U' \cup \{u\}$ induce a $K_{k,k}$, which is a contradiction.

The proof for DR-bulky edges is symmetric and the proof for C-bulky edges can easily be seen by selecting $S' \subseteq \mathsf{succ}^{C}(\phi(u))$ to be the $k-1$ segments $s' \in S'$ with the smallest $s'.y$. Since every $s' \in S'$ is already contained in $\phi(u)$ on the x-axis, then the y coordinate is the only factor determining which elements of $\mathsf{succ}^{C}(\phi(u))$ are contained within a bottomless rectangle that contains $\phi(u)$. ◀

▶ **Lemma 17.** *For each $v \in V$, there are at most $6(k-1)$ thin edges incident to v.*

Proof. For a partial order $\mathscr{P}$, we use $E(\mathscr{P})$ to denote the number of comparable pairs in $\mathscr{P}$.

Let $T \subset N_G(v)$ be the set of nodes u, such that $\{u, v\}$ is thin. Let $\mathscr{P}_l, \mathscr{P}_r, \mathscr{P}_c$ be the partial orders $\mathscr{P}^{DL}, \mathscr{P}^{DR}, \mathscr{P}^{C}$ induced on set $\phi(T)$. According to Observation 15, each pair of segments $s, s' \in \phi(T)$, is comparable in at least one of the partial orders. We then have $|E(\mathscr{P}_l) \cup E(\mathscr{P}_r) \cup E(\mathscr{P}_c)| \geq \binom{|T|}{2}$.

Since for all $u \in T$, the edge $\{u, v\}$ is thin, there are at most $(k-2)$ DL-successors of $\phi(u)$ in $\phi(T)$, so we have $|E(\mathscr{P}_l)| \leq (k-2)|T|$. Similarly, we have at most $(k-2)$ C-successors and $(k-2)$ DR-successors in $\phi(T)$. Therefore, $|E(\mathscr{P}_r)|, |E(\mathscr{P}_c)| \leq (k-2)|T|$. Combining the edges in the three partial orders, we get $|E(\mathscr{P}_l) \cup E(\mathscr{P}_r) \cup E(\mathscr{P}_c)| \leq 3(k-2)|T|$. Putting these inequalities together, we have

$$|T|(|T|-1)/2 = \binom{|T|}{2} \leq 3(k-2)|T|$$

which implies that $|T| \leq 6k - 11 < 6(k-1)$. ◀

▶ **Theorem 18.** *For all $n, m, k \in \mathbb{N}$, $Z_{\mathsf{CHAIN}^3}(m, n; k, k) \leq (3m + 6n)(k-1)$.*

Proof. Let $G = (U \cup V, E) \in \mathsf{CHAIN}^3$ be a $K_{k,k}$-free graph, where $|U| = m$, $|V| = n$. Note that we defined every edge in E as either thin or bulky. According to Lemma 16, the number of bulky edges in E is at most $3(k-1)m$. According to Lemma 17, the number of thin edges in E is at most $6(k-1)n$. This means that the total number of edges is $|E| \leq (3m + 6n)(k-1)$. ◀

▶ **Corollary 19.** *When $m = n$, we have $Z_{\mathsf{CHAIN}^3}(n; k) \leq 9(k-1)n$.*

▶ **Theorem 1** (Dichotomy theorem)**.** *The following dichotomy results hold for all $k \in \mathbb{N}$:*

- $Z_{\mathsf{CHAIN}^3}(n; k) \leq 9n(k-1)$.
- $Z_{\mathsf{CHAIN}^4}(n; k) \in \Omega(nk \cdot \frac{\log n}{\log \log n})$.

Proof. The first part will be proved in Section 4. The second part is a simple corollary of existing results: Chazelle [9] constructed $K_{2,2}$-free PRIG graphs on n' vertices that have at least $\Omega(n' \cdot \frac{\log n'}{\log \log n'})$ edges. Chan and Har-Peled [4] assert the applicability of this result in graph theory, since the original context was pointer machines. By a simple modification that we show in Lemma 36, there exist $K_{k,k}$-free PRIG graphs on N vertices with at least $\Omega(N \cdot (k-1) \cdot \frac{\log \frac{N}{k-1}}{\log \log \frac{N}{k-1}}) = \Omega(N \cdot k \cdot \frac{\log N}{\log \log N})$ edges. Since $\mathsf{PRIG} \subseteq \mathsf{CONV}^2$ (Lemma 31), which is contained in CHAIN^4 [7], then the constructed graphs give a lower bound for Zarankiewicz numbers for CHAIN^4 graphs. ◀

5 Grid Intersection Graphs

This section will use a more formal definition of grid intersection graphs. We say that $(\mathbf{X}, \mathbf{Y}, \phi)$ is a grid intersection representation of a bipartite graph $G = (U \cup V, E)$, if $\mathbf{X}$, $\mathbf{Y}$ are sets of horizontal and vertical segments in $\mathbb{R}^2$ respectively, $\phi : U \mapsto \mathbf{X}, V \mapsto \mathbf{Y}$, and for all $u \in U$, $v \in V$, we have $\{u, v\} \in E$ if and only if $\phi(u)$ and $\phi(v)$ intersect. A bipartite graph G is a grid intersection graph (GIG) if it has a grid intersection representation. For a segment $z \in \mathbf{X} \cup \mathbf{Y}$, let $N(z) = \{z' \in \mathbf{X} \cup \mathbf{Y} : z' \cap z \neq \emptyset\}$, e.g the set of segments that intersect z.

Let k be the minimum positive integer such that G does not have $K_{k,k}$ as a subgraph.

▶ **Observation 20.** *If $G = (U \cup V, E) \in \mathsf{GIG}$ is $27(k-1)$-degenerate, then $|E| \leq 27(k-1)(|U| + |V|)$.*

The proof is done via charging arguments.

5.1 Charging Arguments

For each segment $z \in \mathbf{X} \cup \mathbf{Y}$, let $z.x$ and $z.y$ denote its projections to the x-axis (horizontal) and y-axis (vertical) respectively. Note that the projections are closed intervals and points. For an interval $[a, b]$, let $\min([a, b]) = a$ and $\max([a, b]) = b$. For two intervals I and I', we say that $I < I'$, if $\max(I) < \min(I')$.

For any segment $z \in \mathbf{X} \cup \mathbf{Y}$, we define $\mathrm{DIR}_{down}(z) = \{z' \in \mathbf{X} \cup \mathbf{Y} : z'.y < z.y\}$, i.e. the set of segments whose projection to the y axis is entirely smaller than that of z. Similarly, we define $\mathrm{DIR}_{up}(z) = \{z' \in \mathbf{X} \cup \mathbf{Y} : z'.y > z.y\}$, $\mathrm{DIR}_{left}(z) = \{z' \in \mathbf{X} \cup \mathbf{Y} : z'.x < z.x\}$, $\mathrm{DIR}_{right}(z) = \{z' \in \mathbf{X} \cup \mathbf{Y} : z'.x > z.x\}$. For a set S, we define $\mathrm{DIR}_{up}(S) = \bigcap_{z \in S} \mathrm{DIR}_{up}(z)$ and similarly for the other directions.

▶ **Definition 21.** *We say that a vertical segment $\nu \in \mathbf{Y}$ is **down-heavy** with respect to $\sigma \in \mathbf{X}$, if $\sigma.y \subset \nu.y$ and $|N(\nu) \cap \mathrm{DIR}_{down}(\sigma)| \geq 3(k-1)$.*

▶ **Definition 22.** *We say that a vertical segment $\nu \in \mathbf{Y}$ is **up-heavy** with respect to $\sigma \in \mathbf{X}$, if $\sigma.y \subset \nu.y$ and $|N(\nu) \cap \mathrm{DIR}_{up}(\sigma)| \geq 3(k-1)$.*

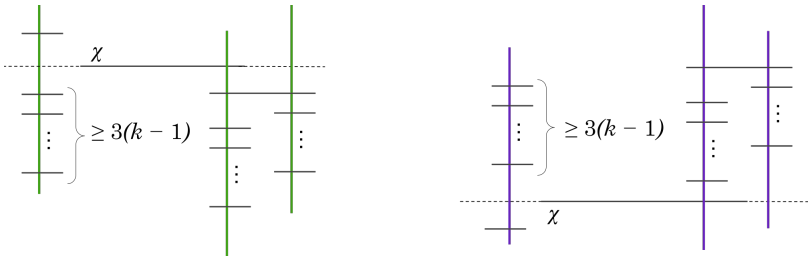


Figure 6 down-heavy segments are represented by the green vertical segments on the left and up-heavy segments by the purple ones on the right.

To show that a node $w \in U \cup V$, such that $|N(w)| \leq 27(k-1)$ exists in any GIG graph, we present a payment scheme algorithm whereby each horizontal segment starts with $27(k-1)$ credits, which is subsequently paid to vertical segments. We will see that each vertical segment $\nu \in \mathbf{Y}$ whose degree is at least $27(k-1)$, receives at least $|N(\nu)|$ credits.

Algorithm 1 Neighbor payments.

From each $\sigma \in \mathbf{X}$
  Pay $\frac{9}{2}$ credits to each of the following vertical segments in $N(\sigma)$:
    $k-1$ leftmost up-heavy vertical segments with respect to σ
    $k-1$ rightmost up-heavy vertical segments w.r.t. σ
    $k-1$ leftmost down-heavy vertical segments w.r.t. σ
    $k-1$ rightmost down-heavy vertical segments w.r.t. σ

Algorithm 2 Further payments.

From each $\sigma \in \mathbf{X}$
  Pay $\frac{9}{4}$ credits to each of the following segments ν, where $\sigma.y \subset \nu.y$:
    $k-1$ rightmost up-heavy vertical segments in $\mathrm{DIR}_{left}(\sigma)$
    $k-1$ rightmost down-heavy vertical segments in $\mathrm{DIR}_{left}(\sigma)$
    $k-1$ leftmost up-heavy vertical segments in $\mathrm{DIR}_{right}(\sigma)$
    $k-1$ leftmost down-heavy vertical segments in $\mathrm{DIR}_{right}(\sigma)$

Now, we will analyze the credits received by an arbitrary vertical segment $\nu \in \mathbf{Y}$.

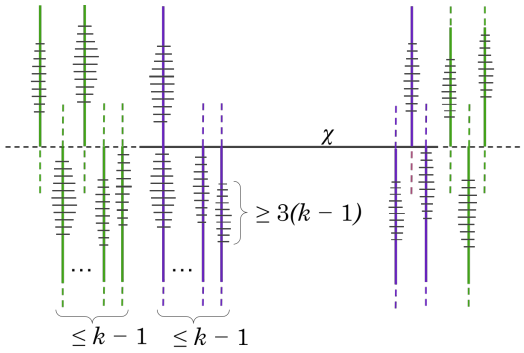


Figure 7 Algorithm 1 credit receivers are shown in purple and Algorithm 2 credit receivers in green.

For a set of horizontal segments $S \subseteq \mathbf{X}$, let $S.y$ denote the interval $[min, max]$, where $min = \min_{\sigma \in S}(\sigma.y)$ and $max = \max_{\sigma \in S}(\sigma.y)$.

▶ **Lemma 23.** *Let $S \subset N(\nu)$, such that*

- $|S| = 3(k-1)$
- $|N(\nu) \cap \mathrm{DIR}_{up}(S)| \geq 3(k-1)$ *(Intuition: S is not among the highest elements of $N(\nu)$)*
- $|N(\nu) \cap \mathrm{DIR}_{down}(S)| \geq 3(k-1)$ *(Intuition: S is not among the lowest elements of $N(\nu)$)*

Then at least $\frac{9}{2}(k-1)$ credits are paid to ν from horizontal segments $\sigma \in \mathbf{X}$, such that $\sigma.y \in S.y$.

The proof is deferred to *Appendix D*.

▶ **Lemma 24.** *If $|N(\nu)| \geq 27(k-1)$, then ν receives at least $|N(\nu)|$ credits.*

Proof. We choose the maximum number ℓ of disjoint sets $S_1, \ldots S_\ell$ in $N(\nu)$, such that every set S_i satisfies the conditions of Lemma 23. Since each set receives $\frac{9}{2}(k-1)$ credits from a distinct set of horizontal segments, then the total amount in credits ν receives is at least $\frac{9}{2}(k-1)\ell$.

It remains to see that $\frac{9}{2}(k-1)\ell \geq |N(\nu)|$. Note that $\ell \geq \lfloor \frac{|N(\nu)|-3(k-1)-3(k-1)}{3(k-1)} \rfloor \geq \frac{|N(\nu)|-9(k-1)}{3(k-1)}$. We get that $\frac{9}{2}(k-1)\ell \geq \frac{9}{2}(k-1)\frac{|N(\nu)|-9(k-1)}{3(k-1)} = \frac{3}{2}|N(\nu)| - \frac{27}{2}(k-1) = |N(\nu)| + \frac{1}{2}(|N(\nu)| - 27(k-1))$. Since according to our assumption $|N(\nu)| \geq 27(k-1)$, then this result is at least $|N(\nu)|$. ◀

▶ **Lemma 25.** *Any $G = (U \cup V, E) \in \mathsf{GIG}$ is $27(k-1)$-degenerate.*

Proof. Assume by contradiction that there is a graph $G = (U \cup V, E) \in \mathsf{GIG}$, where no such node exists. Since according to our assumption, for any node $w \in U \cup V$, we have $|N(w)| \geq 27(k-1)+1$, then by running Algorithm 1 and Algorithm 2, we see that for each horizontal node, fewer credits are given out than their degree. This implies that the amount in credits given by these algorithms is smaller than $|E|$.

For each vertical node $\nu \in \mathbf{Y}$, on the other hand, according to Lemma 24, the amount in credits received is at least that of their degree. This implies that the total amount in credits received from these algorithms greater than or equal to $|E|$. This is a contradiction. ◀

▶ **Theorem 26.** *For all $n, m, k \in \mathbb{N}$, $Z_{\mathsf{GIG}}(m, n; k, k) \leq 27(m+n)(k-1)$.*

Proof. The result follows from Lemma 25 and Observation 7. ◀

▶ **Theorem 3.** *For all* $k, n \in \mathbb{N}$, $Z_{\mathsf{GIG}}(n;k) \leq 54n(k-1)$.

Proof. The result follows from Theorem 26, when $n = m$ ◀

References

1 Stephen J. Bellantoni, Irith Ben-Arroyo Hartman, Teresa M. Przytycka, and Sue Whitesides. Grid intersection graphs and boxicity. *Discret. Math.*, 114(1-3):41–49, 1993. doi:10.1016/0012-365X(93)90354-V.

2 Tom Bohman and Peter Keevash. The early evolution of the H-free process. *Inventiones Mathematicae*, 181(2):291–336, May 2010. doi:10.1007/s00222-010-0247-x.

3 Romain Bourneuf, Matija Bucić, Linda Cook, and James Davies. On polynomial degree-boundedness. *Advances in Combinatorics*, 2024. doi:10.19086/aic.2024.5.

4 Timothy M. Chan and Sariel Har-Peled. On the number of incidences when avoiding an induced biclique in geometric settings. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 1398–1413. SIAM, 2023. doi:10.1137/1.9781611977554.CH50.

5 Timothy M. Chan, Chaya Keller, and Shakhar Smorodinsky. On Zarankiewicz's problem for intersection hypergraphs of geometric objects. In Oswin Aichholzer and Haitao Wang, editors, *41st International Symposium on Computational Geometry, SoCG 2025, June 23-27, 2025, Kanazawa, Japan*, volume 332 of *LIPIcs*, pages 33:1–33:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPICS.SOCG.2025.33.

6 L. Sunil Chandran, Mathew C. Francis, and Rogers Mathew. Chordal bipartite graphs with high boxicity. *Graphs Comb.*, 27(3):353–362, 2011. doi:10.1007/S00373-011-1017-2.

7 Steven Chaplick, Pavol Hell, Yota Otachi, Toshiki Saitoh, and Ryuhei Uehara. Intersection dimension of bipartite graphs. In T. V. Gopal, Manindra Agrawal, Angsheng Li, and S. Barry Cooper, editors, *Theory and Applications of Models of Computation - 11th Annual Conference, TAMC 2014, Chennai, India, April 11-13, 2014. Proceedings*, volume 8402 of *Lecture Notes in Computer Science*, pages 323–340. Springer, 2014. doi:10.1007/978-3-319-06089-7_23.

8 Steven Chaplick, Pavol Hell, Yota Otachi, Toshiki Saitoh, and Ryuhei Uehara. Ferrers dimension of grid intersection graphs. *Discret. Appl. Math.*, 216:130–135, 2017. doi:10.1016/J.DAM.2015.05.035.

9 Bernard Chazelle. Lower bounds for orthogonal range searching: I. the reporting case. *J. ACM*, 37(2):200–212, 1990. doi:10.1145/77600.77614.

10 Thao Do. Representation complexities of semialgebraic graphs. *SIAM J. Discret. Math.*, 33(4):1864–1877, 2019. doi:10.1137/18M1221606.

11 Martin Farber. Characterizations of strongly chordal graphs. *Discret. Math.*, 43(2-3):173–189, 1983. doi:10.1016/0012-365X(83)90154-1.

12 Jacob Fox and János Pach. Separator theorems and Turán-type results for planar intersection graphs. *Advances in Mathematics*, 219(3):1070–1080, 2008. doi:10.1016/j.aim.2008.06.002.

13 Jacob Fox and János Pach. A separator theorem for string graphs and its applications. *Comb. Probab. Comput.*, 19(3):371–390, 2010. doi:10.1017/S0963548309990459.

14 Jacob Fox, János Pach, Adam Sheffer, Andrew Suk, and Joshua Zahl. A semi-algebraic version of Zarankiewicz's problem. *Journal of the European Mathematical Society (EMS Publishing)*, 19(6):1785–1810, 2017. doi:10.4171/JEMS/705.

15 Petr A. Golovach, Pinar Heggernes, Mamadou Moustapha Kanté, Dieter Kratsch, and Yngve Villanger. Enumerating minimal dominating sets in chordal bipartite graphs. *Discret. Appl. Math.*, 199:30–36, 2016. doi:10.1016/J.DAM.2014.12.010.

16 Irith Ben-Arroyo Hartman, Ilan Newman, and Ran Ziv. On grid intersection graphs. *Discret. Math.*, 87(1):41–52, 1991. doi:10.1016/0012-365X(91)90069-E.

17 Oliver Janzer and Cosmin Pohoata. On the Zarankiewicz problem for graphs with bounded VC-dimension. *Comb.*, 44(4):839–848, 2024. doi:10.1007/S00493-024-00095-2.

18 Chaya Keller and Shakhar Smorodinsky. Zarankiewicz's problem via ϵ-t-nets. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th International Symposium on Computational Geometry, SoCG 2024, June 11-14, 2024, Athens, Greece*, volume 293 of *LIPIcs*, pages 66:1–66:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.SOCG.2024.66.

19 Bettina Klinz, Rüdiger Rudolf, and Gerhard J. Woeginger. Permuting matrices to avoid forbidden submatrices. *Discret. Appl. Math.*, 60(1-3):223–248, 1995. doi:10.1016/0166-218X(94)00054-H.

20 Tamás Kővári, Vera T. Sós, and Pál Turán. On a problem of K. Zarankiewicz. *Colloquium Mathematicum*, 3:50–57, 1954. URL: http://eudml.org/doc/210011.

21 Jan Kratochvíl. A special planar satisfiability problem and a consequence of its NP-completeness. *Discret. Appl. Math.*, 52(3):233–252, 1994. doi:10.1016/0166-218X(94)90143-0.

22 Jan Kratochvíl and Jiří Matoušek. NP-hardness results for intersection graphs. *Commentationes Mathematicae Universitatis Carolinae*, 30(4):761–773, 1989. URL: http://eudml.org/doc/17790.

23 Anna Lubiw. Doubly lexical orderings of matrices. In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 396–404. ACM, 1985. doi:10.1145/22145.22189.

24 Haiko Müller. Hamiltonian circuits in chordal bipartite graphs. *Discret. Math.*, 156(1-3):291–298, 1996. doi:10.1016/0012-365X(95)00057-4.

25 Nabil H. Mustafa and János Pach. On the Zarankiewicz problem for intersection hypergraphs. In Emilio Di Giacomo and Anna Lubiw, editors, *Graph Drawing and Network Visualization - 23rd International Symposium, GD 2015, Los Angeles, CA, USA, September 24-26, 2015, Revised Selected Papers*, volume 9411 of *Lecture Notes in Computer Science*, pages 207–216. Springer, 2015. doi:10.1007/978-3-319-27261-0_18.

26 Irina Mustata. *On subclasses of grid intersection graphs.* PhD thesis, Technische Universität Berlin, 2014. doi:10.14279/depositonce-4202.

27 Pranab K. Saha, Asim Basu, Malay K. Sen, and Douglas B. West. Permutation bigraphs and interval containments. *Discret. Appl. Math.*, 175:71–78, 2014. doi:10.1016/J.DAM.2014.05.020.

28 Shakhar Smorodinsky. A survey of Zarankiewicz problem in geometry. *CoRR*, abs/2410.03702, 2024. doi:10.48550/arXiv.2410.03702.

29 István Tomon and Dmitriy Zakharov. Turán-type results for intersection graphs of boxes. *Combinatorics, Probability and Computing*, 30(6):982–987, 2021. doi:10.1017/S0963548321000171.

30 Ryuhei Uehara, Seinosuke Toda, and Takayuki Nagoya. Graph isomorphism completeness for chordal bipartite graphs and strongly chordal graphs. *Discret. Appl. Math.*, 145(3):479–482, 2005. doi:10.1016/J.DAM.2004.06.008.

A Higher Ferrers Dimension

We prove Corollary 2 in this section. Note that the proof closely follows Section 2.2 of Chan and Har-Peled [4]. We repeat their definitions here for completeness. For integers $a \leq b$, we denote $\{a, a+1, \ldots, b\}$ by $[[a : b]]$. A **dyadic range** is an integer range of the form $[[s2^i : (s+1)2^{i+1} - 1]]$ for some integers s and i. Considering a CHAIN graph, we can view the points as integers $\mathcal{P} = \{1, \ldots, n\}$. Denote by $\mathcal{D}(n)$ the set of all dyadic ranges. Each integer (point) appears in at most $\lceil \log n \rceil$ dyadic ranges.

Notice that a ray is simply a range $[[a, n]]$ for some a.

▶ **Lemma 27** (An analogue of Lemma 2.2 in [4]). *Let r be a ray. Denote by $\mathsf{dy}(r)$ the set of minimal disjoint dyadic ranges covering r. Then $\mathsf{dy}(r)$ is unique and contains at most $\lceil \log n \rceil$ ranges.*

Remark that we have $\lceil \log n \rceil$ instead of $2\lceil \log n \rceil$ as in [4] since we have rays instead of intervals. The following lemma implies Corollary 2.

▶ **Lemma 28.** *Let $G = (A \cup B, E) \in \mathsf{CHAIN}^d$ for $d \geq 3$ that does not contain $K_{k,k}$. Then the number of edges in G is at most $O(|A| + |B|) \cdot k \cdot \lceil \log n \rceil^{d-3}$ where $n = \max\{|A|, |B|\}$.*

Proof. The proof is obtained by induction on d. The base case when $d = 3$ is proved in Theorem 18. When $d > 3$, assume the claim is true for all dimensions up to $d - 1$. We write $G = G' \cap \widehat{G}$ where $G' \in \mathsf{CHAIN}$ and $\widehat{G} \in \mathsf{CHAIN}^{d-1}$. Since $G' \in \mathsf{CHAIN}$, it is an intersection bigraph of rays $\mathcal{R}$ and points $\mathcal{P}$ where $a \in A$ is mapped to $r_a \in \mathcal{R}$ and $b \in B$ to $p_b \in \mathcal{P}$. Assume w.l.o.g. that $\mathcal{P} = \{1, \ldots, q\}$ for $q \leq n$.

For each dyadic range $\nu \in \mathcal{D}(q)$, we have an auxiliary graph G_ν which is an induced subgraph $G[A_\nu \cup B_\nu]$ where $A_\nu = \{a \in A : \nu \in \mathsf{dy}(\mathsf{r_a})\}$ and $B_\nu = \{b : p_b \in \nu\}$. Notice that $|E(G)| \leq \sum_{\nu \in \mathcal{D}(n)} |E(G_\nu)|$.

Moreover, $G_\nu = G'[A_\nu \cup B_\nu] \cap \widehat{G}[A_\nu \cup B_\nu]$ where $G'[A_\nu \cup B_\nu]$ is a biclique; therefore $G_\nu \in \mathsf{CHAIN}^{d-1}$ and is free of k-by-k biclique. This allows us to invoke the induction hypothesis, which implies that

$$|E(G)| \leq \sum_{\nu \in \mathcal{D}(n)} O(|A_\nu| + |B_\nu|)k\lceil \log n \rceil^{d-4}$$

Finally, since each vertex $a \in A$ appears in at most $\lceil \log q \rceil \leq \lceil \log n \rceil$ sets A_ν (and similarly for each vertex $b \in B$), this implies that the above sum is at most $O(|A| + |B|)k\lceil \log n \rceil^{d-3}$. ◀

B Intersection of Two Convex Graphs

In this section, we will use a more formal definition of PRIG graphs. We say that $(\mathbf{X}, \mathbf{Y}, \phi)$ is a point-rectangle intersection representation of a graph $G = (U \cup V, E)$, if $\mathbf{X}$, $\mathbf{Y}$ are sets of points and rectangles, respectively, in $\mathbb{R}^2$, $\phi : U \mapsto \mathbf{X}, V \mapsto \mathbf{Y}$, and for all $u \in U$, $v \in V$, we have $\{u, v\} \in E$ if and only if $\phi(u)$ and $\phi(v)$ intersect. A graph $G = (U \cup V, E)$ is a point-rectangle intersection bigraph, if it has a point-rectangle intersection representation. We call the class of such graphs PRIG.

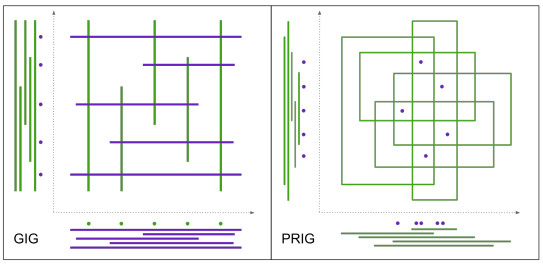


Figure 8 Examples of GIG and PRIG graphs with their projections into axes.

▶ **Lemma 29.** *Any graph $G = (U \cup V, E) \in \mathsf{CONV}^2$ is a disjoint union of* GIG *and* PRIG *graphs.*

Proof. Since $G \in \mathsf{CONV}^2$, then there exist graphs G_1 and G_2, such that $G_1 \cap G_2 = G$. Let V_1 and V_2 be the partitions that the graphs G_1 and G_2, respectively, are convex over, and $U_1 \subseteq V(G_1)$, $U_2 \subseteq V(G_2)$, the other partitions.

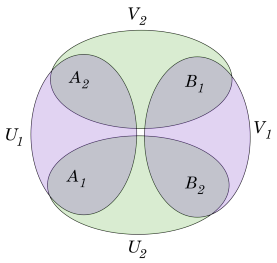


Figure 9 The partitions of two intersecting graphs.

Since CONV graphs are closed under taking induced subgraphs, we can assume that $U_1 \cup V_1 = U_2 \cup V_2$. Let $A_1 = U_1 \cap U_2$, $A_2 = U_1 \cap V_2$, $B_1 = V_1 \cap V_2$, $B_2 = V_1 \cap U_2$, as shown on Figure 9.

We have $U \cup V = A_1 \cup A_2 \cup B_1 \cup B_2$ and all edges must start and end at these nodes. Since A_1 is in the same partition as A_2, and B_2, in at least one of the graphs, there can be no edges between A_1 and $A_2 \cup B_2$. The same is true for B_1. Clearly, G can have one component consisting of nodes $A_1 \cup B_1$ and another component consisting of nodes $A_2 \cup B_2$, with no edges between these components.

It remains to see that $G[A_1 \cup B_1] \in$ PRIG and $G[A_2 \cup B_2] \in$ GIG. To do this, we show that there exist representations of these graphs in PRIG and GIG respectively, such that the projections of these representations to x and y axis are the convex representations of graphs G_1 and G_2 induced on $A_1 \cup B_1$ and $A_2 \cup B_2$ respectively.

Let $H_1 = G_1[A_1 \cup B_1]$ and $H_2 = G_2[A_1 \cup B_1]$. Since G_1 and G_2 are convex over V_1 and V_2, and $B_1 = V_1 \cap V_2$, then H_1 and H_2 are convex over B_1. Let $(\mathbf{X}_1, \mathbf{Y}_1, \phi_1)$ be the point-segment representation of H_1, such that $\mathbf{Y}_1$ are the points. Then $\phi_1 : A_1 \mapsto \mathbf{X}_1, B_1 \mapsto \mathbf{Y}_1$. Let $(\mathbf{X}_2, \mathbf{Y}_2, \phi_2)$ be the point-segment representation of H_2, such that $\mathbf{Y}_2$ are the points. Then $\phi_2 : A_1 \mapsto \mathbf{X}_2, B_1 \mapsto \mathbf{Y}_2$. We construct a mapping $\phi : A_1 \mapsto \mathbf{X}, B_1 \mapsto \mathbf{Y}$, where $\mathbf{X}$ is a set of rectangles, and $\mathbf{Y}$ a set of points in $\mathbb{R}^2$. For each $w \in A_1 \cup B_1$, $\phi(w)$ is the object whose projection to the x-axis is $\phi_1(w)$ and whose projection to the y-axis is $\phi_2(w)$. Note that for all $a \in A_1$, $\phi(a)$ is a rectangle, and for all $b \in B_1$, $\phi(b)$ is a point. It is easy to verify that $\phi(a)$ intersects $\phi(b)$, if and only if $\phi_1(a)$ intersects $\phi_1(b)$ and $\phi_2(a)$ intersects $\phi_2(b)$, which implies that there is an edge $\{a, b\}$ in the graph represented by ϕ, if and only if $\{a, b\} \in E(H_1) \cap E(H_2)$. This means that $(\mathbf{X}, \mathbf{Y}, \phi)$ is a valid representation of $G[A_1 \cup B_1] \in$ PRIG.

Showing that $G[A_2 \cup B_2] \in$ GIG is similar, but not quite symmetric. Let $H_1 = G_1[A_2 \cup B_2]$ and $H_2 = G_2[A_2 \cup B_2]$. Since G_1 and G_2 are convex over V_1 and V_2, and $A_2 \subseteq V_2$, $B_2 \subseteq V_1$, then H_1 and H_2 are convex over A_2 and B_2 respectively. Let us define $(\mathbf{X}_1, \mathbf{Y}_1, \phi_1)$ to be the point-segment representation of H_1, such that $\mathbf{X}_1$ are the points. Then $\phi_1 : A_2 \mapsto \mathbf{X}_1, B_2 \mapsto \mathbf{Y}_1$. Let $(\mathbf{X}_2, \mathbf{Y}_2, \phi_2)$ be the point-segment representation of H_2, such that $\mathbf{Y}_2$ are the points. Then $\phi_2 : A_2 \mapsto \mathbf{X}_2, B_2 \mapsto \mathbf{Y}_2$. We construct a mapping $\phi : A_2 \mapsto \mathbf{X}, B_2 \mapsto \mathbf{Y}$, where $\mathbf{X}$ is a set of horizontal segments, and $\mathbf{Y}$ is a set of vertical segments in $\mathbb{R}^2$. For each $w \in A_2 \cup B_2$, $\phi(w)$ is the object whose projection to the y-axis is $\phi_1(w)$ and whose projection to the x-axis is $\phi_2(w)$. Note that for all $a \in A_2$, $\phi(a)$ is a horizontal segment and for all $b \in B_1$, $\phi(b)$ is a vertical segment. It is easy to verify that $\phi(a)$ intersects $\phi(b)$, if and only if $\phi_1(a)$ intersects $\phi_1(b)$ and $\phi_2(a)$ intersects $\phi_2(b)$, which implies that there is an edge $\{a, b\}$ in the graph represented by ϕ, if and only if $\{a, b\} \in E(H_1) \cap E(H_2)$. This means that $(\mathbf{X}, \mathbf{Y}, \phi)$ is a valid representation of $G[A_1 \cup B_1] \in$ GIG. ◀

For graphs G_1, G_2, let $G_1 \dot\cup G_2$ to denote the disjoint union of G_1 and G_2.

▶ **Lemma 30.** CONV^2 *is closed under disjoint union.*

Proof. Let G, G' be any disjoint graphs in CONV^2. Then we have graphs $G_1, G_2 \in \mathsf{CONV}$, such that $G_1 \cap G_2 = G$, and graphs $G'_1, G'_2 \in \mathsf{CONV}$, such that $G'_1 \cap G'_2 = G'$. Since CONV is closed under disjoint union, then $G_1 \dot\cup G'_1 \in \mathsf{CONV}$ and $G_2 \dot\cup G'_2 \in \mathsf{CONV}$. Then $(G_1 \dot\cup G'_1) \cap (G_2 \dot\cup G'_2) \in \mathsf{CONV}^2$. Since G_1 only shares vertices with G_2, and similarly, G'_1 only shares vertices with G'_2, then $(G_1 \dot\cup G'_1) \cap (G_2 \dot\cup G'_2) = (G_1 \cap G_2) \dot\cup (G'_1 \cap G'_2) = G \dot\cup G' \in \mathsf{CONV}^2$. ◀

▶ **Lemma 31.** $\mathsf{PRIG} \subseteq \mathsf{CONV}^2$.

Proof. Let $G = (U \cup V, E)$ be any graph in PRIG, and $(\mathbf{X}, \mathbf{Y}, \phi)$ the representation of G in PRIG, such that $\mathbf{X}$ is a set of points and $\mathbf{Y}$ a set of rectangles in $\mathbb{R}^2$. Then we can construct mappings ϕ_1 and ϕ_2, such that for all $w \in V(G)$, $\phi_1(w)$ and $\phi_2(w)$ are the projections of $\phi(w)$ to x and y axis respectively. Let G_1 and G_2 be the graphs represented by $\phi_1(w)$ and $\phi_2(w)$ respectively. Since $\phi_1(w)$ maps the vertices $V(G)$ to points and segments based on their partitions, then $G_1 \in \mathsf{CONV}$. Similarly for G_2. Since for any $u \in U$, $v \in V$, we have that $\phi(u)$ intersects $\phi(v)$ if and only if $\phi_1(u)$ intersects $\phi_1(v)$ and $\phi_2(u)$ intersects $\phi_2(v)$, then $G_1 \cap G_2 = G$. This proves that $G \in \mathsf{CONV}^2$. ◀

▶ **Lemma 32.** $\mathsf{GIG} \subseteq \mathsf{CONV}^2$.

Proof. The proof is symmetric to Lemma 31. ◀

Theorem 6: Any graph $G = (U \cup V, E) \in \mathsf{CONV}^2$ if and only if it is a disjoint union of GIG and PRIG graphs.

This follows from *Lemma* 29, *Lemma* 30, *Lemma* 31, and *Lemma* 32.

C Segment Bottomless Rectangle Containment Bigraph

We say that $(\mathbf{X}, \mathbf{Y}, \phi)$ is an interval containment representation of a graph $G = (U \cup V, E)$, if $\mathbf{X}$, $\mathbf{Y}$ in $\mathbb{R}$ are intervals, $\phi : U \mapsto \mathbf{X}, V \mapsto \mathbf{Y}$, and for all $u \in U$, $v \in V$, we have $\{u, v\} \in E$ if and only if $\phi(u)$ is contained within $\phi(v)$. A graph G is an interval containment bigraph, if it has an interval containment representation. According to [7], the class of interval containment bigraphs is CHAIN^2.

▶ **Proposition 33.** *If a graph $G = (U \cup V, E) \in \mathsf{CHAIN}$ has a point ray representation $(\mathbf{X}, \mathbf{Y}, \phi)$, $\phi : U \mapsto \mathbf{X}, V \mapsto \mathbf{Y}$, then it also has a point ray representation $(\mathbf{X}', \mathbf{Y}', \phi')$, $\phi' : V \mapsto \mathbf{X}', U \mapsto \mathbf{Y}'$, where $\mathbf{X}, \mathbf{X}'$ are points and $\mathbf{Y}, \mathbf{Y}'$ are rays.*

Proof. We are given $G = (U \cup V, E) \in \mathsf{CHAIN}$ with a point ray representation $(\mathbf{X}, \mathbf{Y}, \phi)$, $\phi : U \mapsto \mathbf{X}, V \mapsto \mathbf{Y}$, where $\mathbf{X}$ represents points and $\mathbf{Y}$ represents rays extending to ∞ in $\mathbb{R}$.

For $\nu \in \mathbf{Y}$, let $\nu.p$ denote the minimum point in ray ν. Then for all $u \in U$, $v \in V$, we say that $\phi(v).p < \phi(u)$, if and only if $\{u, v\} \in E$.

We construct $(\mathbf{X}', \mathbf{Y}', \phi')$ such that for each $u \in U$, $\phi'(u) = (-\infty, \phi(u)]$, $\phi'(v) = \phi(v).p$. Then for all $u \in U$, $v \in V$, $\phi'(u)$ and $phi'(v)$ intersect if $\phi'(v) < \phi'(u).p$, which happens if and only if $\phi(v).p < \phi(u)$. This implies that $(\mathbf{X}', \mathbf{Y}', \phi')$ is a representation of G, where the partition U is mapped to rays and V to points. By flipping the representation, it is easy to see that there is an equivalent representation where rays extend to ∞. ◀

▶ **Lemma 14.** A graph G is of Ferrers dimension three ($G \in \mathsf{CHAIN}^3$) if and only if it is a bottomless rectangle segment containment graph.

Proof. Let $(\mathcal{S}, \mathcal{B}, \phi)$ be a segment bottomless rectangle representation of graph $G = (U \cup V, E)$, such that $\phi : U \mapsto \mathcal{S}, V \mapsto \mathcal{B}$. We will see that $G \in \mathsf{CHAIN}^3$.

First, we consider the graph $G_x = (U \cup V, E_x)$ represented by the projections of $\mathcal{S}, \mathcal{B}$ to the x axis. Since G_x is an interval containment bigraph, then $G_x \in \mathsf{CHAIN}^2$ ([27], [7]). Secondly, consider the graph $G_y = (U \cup V, E_y)$ represented by the projections of $\mathcal{S}, \mathcal{B}$ to the y axis. Since G_y is an intersection graph of points and rays, then $G_y \in \mathsf{CHAIN}$. Since each $b \in \mathcal{B}$ contains $s \in \mathcal{S}$ if and only if $s.x \subseteq b.x$ and $s.y \subseteq b.y$, then $G = G_x \cap G_y \in \mathsf{CHAIN}^2 \cdot \mathsf{CHAIN} = \mathsf{CHAIN}^3$.

Next, we will show that any graph $G = (U \cup V, E) \in \mathsf{CHAIN}^3$ has a segment bottomless rectangle representation. According to [7], there exist graphs $G_x = (U \cup V, E_x) \in \mathsf{CHAIN}^2$ and $G_y = (U \cup V, E_y) \in \mathsf{CHAIN}$, such that $G_x \cap G_y = G$. Let $(\mathbf{X}_x, \mathbf{Y}_x, \phi_x)$ be an interval containment representation of G_x, such that $\mathbf{X}$ represent the contained segments and $\mathbf{Y}$ the containing ones. Let us consider the case that $\phi_x : U \mapsto \mathbf{X}_x, V \mapsto \mathbf{Y}_x$, the case that $\phi_x : V \mapsto \mathbf{X}_x, U \mapsto \mathbf{Y}_x$ is symmetric. Let $(\mathbf{X}_y, \mathbf{Y}_y, \phi_y)$ be a point ray representation of G_y, such that $\mathbf{X}_y$ are points and $\mathbf{Y}_y$ are rays, $\phi_y : U \mapsto \mathbf{X}_y, V \mapsto \mathbf{Y}_y$ (Proposition 33).

We construct a segment bottomless rectangle representation $(\mathcal{S}, \mathcal{B}, \phi)$, $\phi : U \mapsto \mathcal{S}, V \mapsto \mathcal{B}$, such that for all $w \in U \cup V$, $\phi(w).x = \phi_x(w)$, $\phi(w).y = \phi_y(w)$. Let G' be the graph represented by $(\mathcal{S}, \mathcal{B}, \phi)$. For all $u \in U$, $v \in V$, $\phi(u)$ is contained within $\phi(v)$ if and only if $\phi_x(u)$ is contained within $\phi_x(v)$ and $\phi_y(u)$ is contained within $\phi_y(v)$. This implies that $G' = G_x \cap G_y$, which means that $(\mathcal{S}, \mathcal{B}, \phi)$ is a representation of G. ◀

D Grid Intersection Graphs

We say $\sigma \in N(\nu)$ is **generous to** ν (or simply generous), if *Algorithm* 1 pays any credits to ν from σ. We consider the remaining segments in $N(\nu)$ **stingy** to ν.

First, we will see that if for a section of a vertical node, not enough credits are gained from Algorithm 1, then there are horizontal segments in that section, which will contribute to Algorithm 2 instead.

▶ **Lemma 34.** *Let $S \subseteq N(\nu)$, and $S' \subseteq S$ be the stingy subset, such that*

- $|S| \geq 3(k-1)$
- $|S - S'| \leq k - 1$
- $N(\nu) \subseteq \mathrm{DIR}_{up}(S) \cup \mathrm{DIR}_{down}(S) \cup S$ *(S contains consecutive neighbors of ν)*

Then there exists a set $S^* \in \mathrm{DIR}_{right}(\nu)$, such that

- $|S^*| \geq k - 1$
- $S^*.y \subseteq S.y$
- $\bigcap_{\sigma \in S' \cup S^*} \sigma.x \neq \emptyset$

Proof. The number of stingy nodes is at least $2(k-1)$. Let the stingy subset of S be S'. For any $\sigma \in S'$, there are at least $k-1$ segments in either $\mathrm{DIR}_{up}(\sigma) \cap S'$ or $\mathrm{DIR}_{down}(\sigma) \cap S'$.

Let $\sigma \in S'$ be the segment with the smallest $\max(\sigma.x)$. Let $\mathrm{DIR}_{down}(\sigma) \cap S' = S'_{down}$ and $\mathrm{DIR}_{up}(\sigma) \cap S' = S'_{up}$. We consider the case that $|S'_{down}| \geq k-1$. The case that $|S'_{up}| \geq k-1$ is symmetric. Let Q be the $(k-1)$ rightmost down-heavy vertical segments in Algorithm 1, that are each paid $\frac{9}{2}$ credits from σ. Let $\nu' \in Q$ be the vertical segment with the largest $\min(\nu'.y)$.

Since $\nu' \in Q$ has the largest $\min(\nu'.y)$ and $\sigma \in S'$ has the smallest $\max(\sigma.x)$, then $\{\sigma\} \cup S'_{down} \cap N(v')$ and $Q \cup \{\nu\}$ induce a biclique (Figure 10). Since $|Q \cup \{\nu\}| = k$ and there cannot exist a $K_{k,k}$, then $|S'_{down} \cap N(v')| \leq k-2$, which implies that $\min(\nu'.y)$ is greater than $\min(S.y)$ (ν' does not extend far enough down to include $k-1$ stingy nodes below σ).

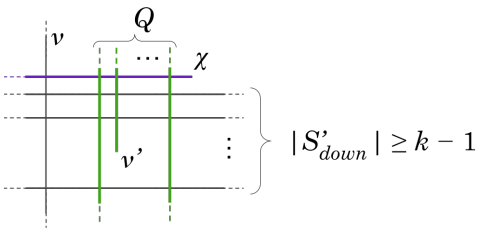


Figure 10 Depicted are the stingy horizontal segments below ν, the vertical segments Q (green), and ν.

Since ν' is down-heavy w.r.t σ, then $|N(\nu') \cap \mathrm{DIR}_{down}(\sigma)| \geq 3(k-1)$ (Definition 21). Let $\Gamma = N(\nu') \cap \mathrm{DIR}_{down}(\sigma)$. Note that $\Gamma.y \subseteq S.y$, because $\min(\nu'.y)$ is greater than $\min(S.y)$ and $\max(\Gamma.y) < \sigma.y$. Due to how S was chosen (the last condition), this implies that the nodes in Γ that intersect ν are in S.

Since $(k-1)$ nodes of $S'_{down} = \mathrm{DIR}_{down}(\sigma) \cap S'$ cannot be in $N(\nu')$, then $|\Gamma \cap S'| \leq k-2$. Since $|S - S'| \leq k-2$, then $|\Gamma - S| \geq 3(k-1) - (k-2) - (k-2) = k+1$.

Let us consider the conditions for $S^* = \Gamma - S$. Since $\Gamma \in N(\nu')$, $\nu' \in Q \subseteq \mathrm{DIR}_{right}(\nu)$, and none of $\Gamma - S$ intersect ν, then $\Gamma - S \in \mathrm{DIR}_{right}(\nu)$. We already saw that $|\Gamma - S| \geq k-1$ and $\Gamma.y \subseteq S.y$. Since $\sigma \in S'$ is the segment with the smallest $\max(\sigma.x)$, then all intervals in S' projected to the x axis must include $\nu'.x$. We have $\nu'.x \in \bigcap_{\sigma \in S' \cup \Gamma} \sigma.x$, which fulfills the last condition. ◀

▶ **Lemma 35.** *Let $S \subseteq N(\nu)$, and $S' \subseteq S$ be the stingy subset, such that*

- $|S| \geq 3(k-1)$
- $|S - S'| \leq k-1$
- $N(\nu) \subseteq \mathrm{DIR}_{up}(S) \cup \mathrm{DIR}_{down}(S) \cup S$ *(S contains consecutive neighbors of ν)*

Then there exists a set $S^* \in \mathrm{DIR}_{left}(\nu)$, such that

- $|S^*| \geq k-1$
- $S^*.y \subseteq S.y$
- $\bigcap_{\sigma \in S' \cup S^*} \sigma.x \neq \emptyset$

Proof. The proof is symmetric to Lemma 34. ◀

▶ **Lemma 23.** Let $S \subset N(\nu)$, such that

- $|S| \geq 3(k-1)$
- $|N(\nu) \cap \mathrm{DIR}_{up}(S)| \geq 3(k-1)$ (S is not among the highest elements of $N(\nu)$)
- $|N(\nu) \cap \mathrm{DIR}_{down}(S)| \geq 3(k-1)$ (S is not among the lowest elements of $N(\nu)$)
- $N(\nu) \subseteq \mathrm{DIR}_{up}(S) \cup \mathrm{DIR}_{down}(S) \cup S$ (S contains consecutive neighbors of ν)

At least $\frac{9}{2}(k-1)$ credits are paid to ν from horizontal segments $\sigma \in \mathbf{X}$, such that $\sigma.y \in S.y$.

Proof. If S contains at least $(k-1)$ generous nodes, which each contribute at least $\frac{9}{2}$ credits (*Algorithm* 1), the lemma holds. Otherwise, let S' be the set of stingy nodes in S. We have that $|S - S'| \leq k-2$.

Let $R \in \mathrm{DIR}_{right}(\nu)$ be the set of nodes such that $R.y \subseteq S.y$ and for each $\sigma' \in R$, $\sigma'.x$ intersects $\bigcap_{\sigma \in S'} \sigma.x$. According to Lemma 34, $|R| \geq k-1$. Let $\hat{R}$ be the $k-1$ segments $\sigma \in R$ with the smallest $\min(\sigma.x)$.

We will see that Algorithm 2 pays $\frac{9}{4}$ credits to ν from every segment $\sigma \in \hat{R}$. Let $\sigma \in \hat{R}$ be an arbitrary segment. Let $\mathrm{DIR}_{down}(\sigma) \cap S' = S'_{down}$ and $\mathrm{DIR}_{up}(\sigma) \cap S' = S'_{up}$. Since $|S'| \geq 3(k-1) - (k-2) = 2k-1$, then either $|S'_{down}| \geq k$ or $|S'_{up}| \geq k$. We consider the case that $|S'_{down}| \geq k$. The case that $|S'_{up}| \geq k$ is symmetric. Let $Q \subseteq \mathrm{DIR}_{left}(\sigma)$ be the $(k-1)$ rightmost down-heavy vertical segments in Algorithm 2, that are each paid $\frac{9}{4}$ credits from σ. Note that if $\nu \in Q$, then we have shown that Algorithm 2 pays $\frac{9}{4}$ credits to ν from σ. Since, according to our choice of S, it is not among the lowest neighbors of $N(\nu)$, then ν is down-heavy concerning σ. This implies that $\nu \in Q$, unless all nodes of Q are to the right of ν. Since we have reached our goal in the case of $\nu \in Q$, let us now consider only the case that $Q \in \mathrm{DIR}_{right}(\nu)$ (Figure 11). Let $\nu' \in Q$ be the vertical segment with the largest $\min(\nu'.y)$.

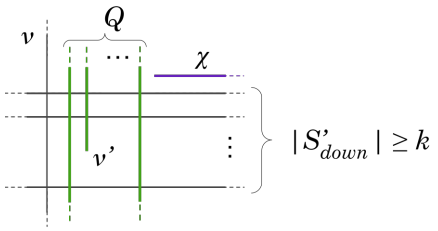


Figure 11 Depicted are the stingy horizontal segments below ν, the vertical segments Q (green), and ν.

Since ν and some point in $\sigma.x$ must intersect $\bigcap_{\sigma' \in S'_{down}} \sigma'.x$ on the x axis, then all segments in Q (which are between them) also intersect all segments in S' when projected to the x-axis. Since $\nu' \in Q$ has the largest $\min(\nu'.y)$, then $S'_{down} \cap N(v')$ and $Q \cup \{\nu\}$ induce a biclique (Figure 11). Since $|Q \cup \{\nu\}| = k$ and there cannot exist a $K_{k,k}$, then $|S'_{down} \cap N(v')| \leq k-1$, which implies that $\min(\nu'.y)$ is greater than $\min(S.y)$ (ν' does not extend far enough down to include k stingy nodes below σ).

Since ν' is down-heavy w.r.t σ, then $|N(\nu') \cap \mathrm{DIR}_{down}(\sigma)| \geq 3(k-1)$ (Definition 21). Let $\Gamma = N(\nu') \cap \mathrm{DIR}_{down}(\sigma)$. Note that $\Gamma.y \subseteq S.y$, because $\min(\nu'.y)$ is greater than $\min(S.y)$ and $\max(\Gamma.y) < \sigma.y$. Due to how S was chosen (the last condition), this implies that the nodes in Γ that intersect ν are in S. Since k nodes of $S'_{down} = \mathrm{DIR}_{down}(\sigma) \cap S'$ cannot be in $N(\nu')$, then $|\Gamma \cap S'| \leq k-1$. Since $|S - S'| \leq k-2$, then $|\Gamma - S| \geq 3(k-1) - (k-1) - (k-2) = k$.

Since all nodes that intersect ν in Γ, must be in S, then $\Gamma - S = \Gamma \cap \mathrm{DIR}_{right}(\nu) \cup \Gamma \cap \mathrm{DIR}_{left}(\nu)$. Since, according to our assumption, $\nu' \in Q \subseteq \mathrm{DIR}_{right}(\nu)$, then no horizontal segment in $\Gamma \in N(\nu')$ can be to the left of ν without intersecting ν. Let $\hat{\Gamma} = (\Gamma - S) \cap \mathrm{DIR}_{right}(\nu)$. Then we have $|\hat{\Gamma}| \geq k$.

As discussed, when projected to the x-axis, all segments in Q intersect all segments in S'. We have $\nu' \in Q$ and $\Gamma \in N(\nu')$. Therefore, on the x-axis, we have a point $\nu'.x$ that intersects S' as well as Γ. According to how we defined R, we have $\hat{\Gamma} \subseteq R$.

Since for all $\sigma^* \in \Gamma$, $\min(\sigma^*.x) \leq v'.x < \min(\sigma.x)$, then $\hat{R}$ contains $\hat{\Gamma}$, if it contains σ. This implies that $|\hat{R}| \geq k$, which is a contradiction, since we chose $\hat{R}$ such that it would only contain $k-1$ segments. This implies that the case that $v \notin Q$ is impossible.

Let $L \in \mathrm{DIR}_{left}(\nu)$ be the set of nodes such that $L.y \subseteq S.y$ and for each $\sigma' \in L$, $\sigma'.x$ intersects $\bigcap_{\sigma \in S'} \sigma.x$. According to Lemma 35, $|L| \geq k-1$. Let $\hat{L}$ be the $k-1$ segments $\sigma \in L$ with the largest $\max(\sigma.x)$. The proof that Algorithm 2 pays $\frac{9}{4}$ credits to ν from every segment $\sigma \in \hat{L}$ is symmetric. ◀

E Lower bounds

▶ **Lemma 36.** *If there exists a $K_{2,2}$-free* PRIG *graph G on n nodes and m edges, then there exists a $K_{k,k}$-free* PRIG *graph G' on $(k-1)\cdot n$ nodes and $(k-1)^2\cdot m$ edges.*

Proof. Let $G=(U\cup V,E)$ be any $K_{2,2}$-free graph in PRIG, and $(\mathbf{X},\mathbf{Y},\phi)$ the representation of G in PRIG, such that $\phi: U\mapsto \mathbf{X}, V\mapsto \mathbf{Y}$, $\mathbf{X}$ is a set of points and $\mathbf{Y}$ a set of rectangles in $\mathbb{R}^2$.

We construct a new graph $G=(U'\cup V',E')$, that is represented in PRIG by $(\mathbf{X}',\mathbf{Y}',\phi')$, $\phi: U'\mapsto \mathbf{X}', V'\mapsto \mathbf{Y}'$ as follows. For every $\sigma\in\mathbf{X}$, we create $k-1$ copies in $\mathbf{X}'$, and for every $\nu\in\mathbf{Y}$, we create $k-1$ copies in $\mathbf{Y}'$. It is easy to see that $|U'|=(k-1)|U|$, $|V'|=(k-1)|V|$. Let us see the number of edges. For every $u\in U$, $\phi(u)$ intersects with $|N(u)|$ rectangles in G. In G', a copy of $\phi(u)$ intersects with all the copies of $\phi(N(u))$, which number in $(k-1)\cdot|N(u)|$. Let u' be a node in G' that is represented by a copy of $\phi(u)$. The degree of u' is $k-1$ times larger than the degree of u. Since the number of nodes in U' is also $k-1$ times larger than in U, then $|E'|=|E|\cdot(k-1)^2$.

Finally, let us see that G' is $K_{k,k}$-free. Suppose by contradiction that G' has $K_{k,k}$ as a subgraph. Let $S_U\subseteq U'$ and $S_V\subseteq V'$ be the nodes that induce $K_{k,k}$ in G'. Then there must be at least two nodes $u,u^*\in U$, whose copies form S_U, and at least two nodes $v,v^*\in V$ whose copies form S_V. Since $G'[S_U\cup S_V]$ is a biclique, then it has edges between all pairs of nodes. There can be an edge between a pair of nodes $\{u',v'\}$, $u'\in U'$, $v'\in V'$, if and only if that edge existed in G between the nodes whose copies u' and v' are. This implies that $u,u^*\in U$ and $v,v^*\in V$ must have induced a $K_{2,2}$ in G. This is a contradiction. ◀

Similarly, we can construct copies of other geometric intersection graphs.

▶ **Corollary 37.** *If there exists a $K_{2,2}$-free* GIG *graph G on n nodes and m edges, then there exists a $K_{k,k}$-free* GIG *graph G' on $(k-1)\cdot n$ nodes and $(k-1)^2\cdot m$ edges.*

▶ **Lemma 38.** $Z_{\mathsf{CHAIN}}(m,n;k)\geq (m+n)(k-1)-O(k^2)$.

Proof. Consider a chain graph $G=(U\cup V,E)$ where V corresponds to the rightward rays $\{r_1,r_2,\ldots,r_n\}$ and U to the points $\{p_1,p_2,\ldots,p_m\}$. Let all the points be placed on the real line such that each point p_i is at coordinate i. The rays $r_1,\ldots,r_{k-1}$ start at the coordinate 0, which contains all the points. The remaining rays $r_k,\ldots,r_n$ start at coordinate $(m-k+1.5)$, so they contain $k-1$ points (see Figure 12). Clearly, the graph does not contain a biclique $K_{k,k}$. The number of edges is $n(k-1)+(m-k+1)(k-1)=(n+m)(k-1)-(k-1)^2$. ◀

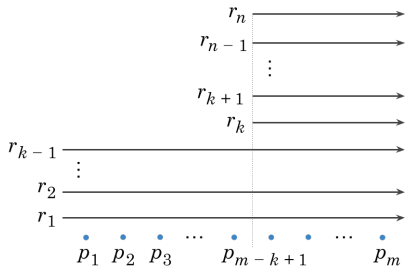


Figure 12 A point-ray representation of the lower bound graph.

E.1 Unit Grid Intersection Graphs

Although we do not know the correct bound for the grid intersection graphs, this section presents a lower bound construction that suggests that the leading constant of $Z(n;k)$ is likely at least 3. This contrasts grid intersection graphs with chordal bigraphs (where the tight bound has a leading constant 2).

Our lower bound holds for the special case where all segments have unit length; such an intersection bigraph is called a Unit Grid Intersection Graph (UGIG).

▶ **Theorem 39.** *We have* $Z(n;k) \geq (k-1)3n - O(k\sqrt{kn})$ *for unit grid intersection graphs.*

First, we construct the graph G' for $Z(n;2) \geq 3n - 2\sqrt{n}$, and then we apply *Corollary* 37 to get a graph for $Z(n;k)$. Note that since the method used in the Lemma is duplication, then the new graph created in the proof will also produce a UGIG. For any $t \in \mathbb{N}$ we can construct a UGIG graph as follows (Figure 13).

Algorithm 3 GIG graph construction.

Create horizontal segments $\mathbf{X}$: **foreach** $i \in [0, t-1], j \in [0, 2t-1]$ **do**
  add $[8i, 8i+7]$ at y-coordinate $4j+1$ to $\mathbf{X}$
  add $[8i-4, 8i+3]$ at y-coordinate $4j+3$ to $\mathbf{X}$
Create vertical segments $\mathbf{Y}$: **foreach** $i, j \in [0, t-1]$ **do**
  add $[8i+2, 8i+8]$ at x-coordinate $8j+1$ to $\mathbf{Y}$
  add $[8i-2, 8i+4]$ at x-coordinate $8j+2$ to $\mathbf{Y}$
  add $[8i, 8i+6]$ at x-coordinate $8j+5$ to $\mathbf{Y}$
  add $[8i+4, 8i+10]$ at x-coordinate $8j+6$ to $\mathbf{Y}$

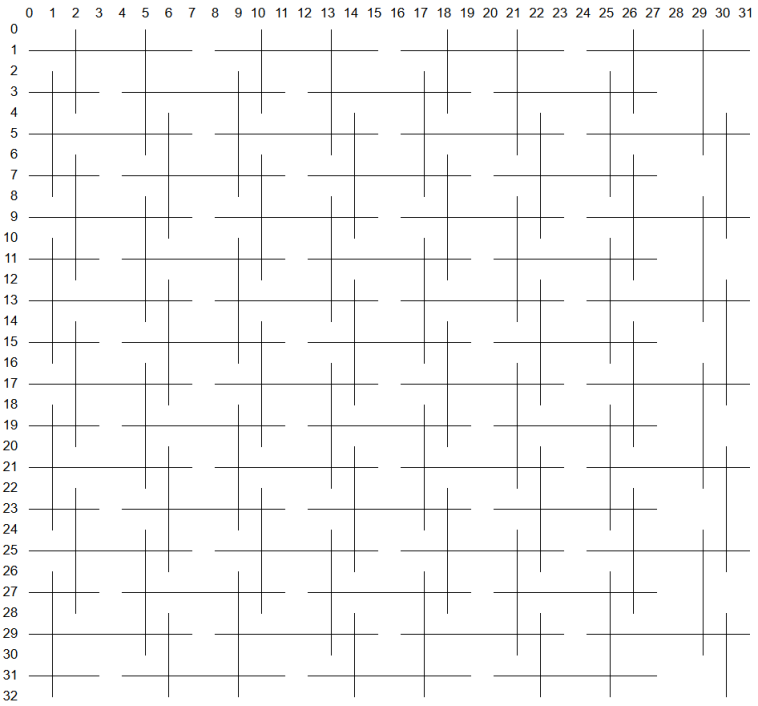


Figure 13 The segments created in Algorithm 3 for $t = 3$. Only the area of the graph with intersections is depicted. (Some segments also have negative coordinates).

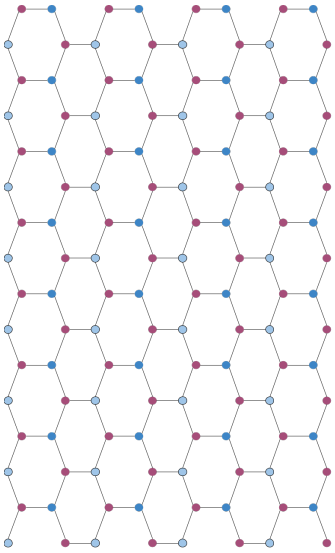


Figure 14 A planar representation of the graph G', where the first and second types of horizontal segments are depicted by blue and light blue nodes, respectively, and red nodes depict vertical segments.

Note that $|\mathbf{X}| = t \cdot 2t + t \cdot 2t = 4t^2$ and $|\mathbf{Y}| = t^2 + t^2 + t^2 + t^2 = 4t^2$. To count the number of edges, we count the edges incident to horizontal segments $\sum_{\sigma \in \mathbf{X}} \deg_{G'}(\sigma)$. Consider the segments that are added by the first line in the algorithm. For these segments, the degree for $j = 0$ (y-coordinate 1) is 2, and for all others, the degree is 3 (Figure 13). The number of edges incident to the first type of horizontal segments is $3 \cdot 2t^2 - t$. Now, consider the segments added by the second line in the algorithm. For these segments, all cases where $i = 0$ have one degree less than 3, as well as the cases where $j = 2t - 1$ (the case where both are true has degree $3 - 2 = 1$, see Figure 13). In total, the number of edges incident to these segments is therefore $3 \cdot 2t^2 - 2t - t$. The constructed graph has $|\mathbf{X}| = |\mathbf{Y}| = 4t^2$ and the number of edges is $\sum_{\sigma \in \mathbf{X}} \deg_{G'}(\sigma) = 12t^2 - 4t$.

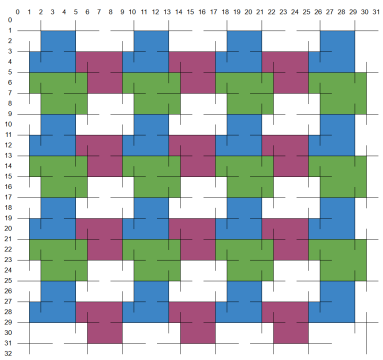


Figure 15 Areas bordered by segments are colored white, blue, green, or red.

Let **faces** be the areas bordered by segments. To see that the graph is $K_{2,2}$-free, note that a GIG representation of a $K_{2,2}$ would form a rectangle. Since segments in each partition have identical lengths, a rectangle that contains only non-rectangle faces cannot exist. Since none of the faces in the graph are rectangles (Figure 15), the graph is $K_{2,2}$-free. This finishes the proof for $Z(n;2) \geq 3n - 2\sqrt{n}$.

Next, we use Corollary 37, to create a $K_{k,k}$-free graph $G = (U \cup V, E)$, where $|U| = |V| = 4(k-1)t^2$ and $|E| = (k-1)^2(12t^2 - 4t)$. Using the value $n = 4(k-1)t^2$, we get $|E| > (k-1)(3n - 2\sqrt{(k-1)n})$.

NNP-NET: Accelerating t-SNE Graph Drawing for Very Large Graphs by Neural Networks

Ilan Hartskeerl
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Tamara Mchedlidze
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Simon van Wageningen
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Peter Vangorp
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Alexandru Telea
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Abstract

tsNET is a recent graph drawing (GD) method that creates high quality layouts but suffers from a very high runtime. We present a new GD method, NNP-NET, which reduces tsNET's time complexity to generate layouts for very large graphs in seconds. Additionally, we extend tsNET to support drawing graphs with edge weights. We accomplish this by replacing tsNET's t-SNE projection with Neural Network Projection (NNP), a fast dimensionality reduction (DR) method that can imitate any given DR method. Our experiments show that NNP-NET gets good quality results when compared to other state-of-the art GD methods while yielding a better computational scalability.

2012 ACM Subject Classification Human-centered computing → Graph drawings; Computing methodologies → Neural networks

Keywords and phrases supervised graph drawing, dimensionality reduction, t-SNE

Digital Object Identifier 10.4230/LIPIcs.GD.2025.22

Supplementary Material *Software (Source Code)*: `https://github.com/IlanHartskeerl/NNP-NET` [26], archived at `swh:1:dir:539da3e3adda73abc8bb1af0c2a5c59fb7abef49`

1 Introduction

Multiple techniques have been proposed to create straight-line drawings of graphs, including classical approaches such as force-directed methods [11, 16], spectral methods [4, 24] and recently machine learning approaches [23, 20, 39]. Dimensionality reduction (DR) methods take a particular approach to GD: They encode the graph structure in a high-dimensional space and map, or project, this data to the 2D space so as to preserve data structure [18, 31, 25]. DR methods offer high flexibility on the information to preserve during projection, can encode node or edge attributes, and are battle-tested in fields such as machine learning and data visualization.

tsNET [30], a relatively recent method in the DR class, uses the well-known t-SNE projection technique [42] to create high-quality graph layouts. Yet, tsNET has quadratic time and space complexity in the number of graph nodes, making it unsuitable for graphs with over a few thousand nodes. While the time complexity can be partially alleviated by GPU-based t-SNE implementations [35, 7], and multilevel schemes and sparse matrices [47] can reduce space complexity, a more radical approach is to entirely *replace* t-SNE. A good candidate here is NNP [14] – a deep-learning projection method that can imitate any DR algorithm,


© Ilan Hartskeerl, Tamara Mchedlidze, Simon van Wageningen, Peter Vangorp, and Alexandru Telea; licensed under Creative Commons License CC-BY 4.0



has linear time complexity in input size, is very simple to use, is robust to small input perturbations, and has out-of-sample ability. Yet, NNP training requires a high-dimensional dataset rather than a distance matrix.

We address this challenge and propose NNP-NET, which replaces tsNET's costly t-SNE step with NNP in a simple and generic fashion, resulting in the following contributions:

Scalability: NNP-NET has a time and space complexity linear in the input graph size. It handles graphs of over 1M nodes in similar and often lower time than DRGraph, the only t-SNE based method we are aware of that can scale to such sizes;

Quality: NNP-NET produces layouts that visually look similar, and have similar quality metrics, to those created by state-of-the art GD methods;

Genericity: NNP-NET can handle any type of graph, including edge-weighted graphs;

Stability: NNP-NET inherits NNP's robustness to noise;

Ease of use: NNP-NET can be used out-of-the-box without tuning any parameters.

The structure of this paper is as follows. Section 2 presents relevant related work. Section 3 details our method. Section 4 compares NNP-NET with state-of-the-art GD methods on a wide range of graphs and using various quality metrics. Section 5 discusses NNP-NET's strengths and limitations. Finally, Sec. 6 concludes the paper.

2 Background and Related Work

Given a graph $G = (V, E)$ with nodes $V = \{v_i\}_{i=1}^N$ and edges $E = \{(v_i, v_j) \in V \times V\}$, a straight-line graph drawing algorithm bijectively maps nodes of V to points $P = \{\mathbf{p}_i\}_{i=1}^N \subset \mathbb{R}^m$, where typically $m = 2$. Optional edge weights $w_{ij} \in \mathbb{R}^+$ can control P by influencing the target edge length $\|\mathbf{p}_i - \mathbf{p}_j\|$. We set $w_{ij} = 1$ in case weights are not given.

2.1 Graph Layout Methods

Hundreds of GD methods have been proposed in the past decades [19, 38]. We next focus on methods that aim to satisfy the contributions outlined in Sec. 1.

Force directed methods. Such methods create a layout by modeling the graph as a physical system where attraction forces describe graph edges and repulsion forces aim to reduce drawing visual clutter [16, 18, 28, 33]. Edge weights can be easily incorporated in the attraction factor. Classical force directed methods have a time complexity of $O(N^2)$, which is slow for large graphs. SFDP [27] and the fast multipole multilevel method (FM^3 [22]) use multilevel schemes to reduce this time complexity: A set of increasingly smaller graphs $\{G^i\}_1^K$ is created from the input graph G; the smallest graph G^K is then laid out using a suitable technique, after which its drawing is used to build the drawing for G^{K-1}, and next for all G^i until G. This yields a time complexity of $O(N \log(N))$, significantly better than the original $O(N^2)$ complexity, yet still below the linear $O(N)$ we aim at.

Dimensionality reduction methods. Dimensionality reduction (DR) and GD have significant overlap, where DR algorithms can be adapted to a GD context [34]. DR based methods encode the graph adjacency information E into a distance matrix and then use classical DR techniques (discussed next in Sec. 2.2) to create the drawing. For example, one can encode the shortest-path distances between all node pairs in the distance matrix and then reduce this to a 2D embedding using classical multidimensional scaling (MDS [40]). However, this

approach has a time and space complexity of $O(N^2)$. PMDS [4] improves by using a smaller matrix of distances from all nodes to a small set of so-called *pivots*. This makes PMDS fast but the quality of the produced drawings falls behind other GD methods. tsNET [30] follows a similar idea but replaces MDS by t-SNE [42], one of the highest-quality existing DR methods [15]. To increase quality and robustness, t-SNE is started from a set of 2D node positions computed using PMDS – a variation called tsNET*. While tsNET(*) layouts are of high quality, the method has a $O(N^2)$ space and time complexity. DRGraph[47] improves upon tsNET to achieve a linear time complexity. A sparse similarity matrix, where only the k nearest neighbors of each node are present, is used. To accelerate t-SNE, a negative sampling technique [32] is used to approximate the gradient of the cost function. A multilevel layout scheme is used to reduce the number of t-SNE iterations. Yet, DRGraph cannot handle weighted graphs and has many complex parameters which make its practical use challenging.

Machine learning methods. SGD^2 [3] improves upon the earlier method GD^2 [2] by using stochastic gradient descent like its predecessor [46]. SGD^2 [3] directly optimizes for quality metrics which allows one to control the desired properties of the resulting drawing. Time complexity depends on the metric optimized for, which in the case for stress is $O(N^2)$. Stress-Plus-X [10] takes a similar approach, optimizing for a specific set of metrics (stress, edge crossings, minimum angle and upwardness). Other recent approaches exploit advances in deep learning to support graph drawing. Yet, several challenges remain here. DNN^2 [20, 21] and DeepGD [43], two recent methods in this area, were only trained to lay out graphs of about 100 nodes due to long training times. DeepDrawing [45] uses a Long Short Term Memory (LSTM) network in order to imitate a ground truth layout. This means that it is only able to create a layout for a graph that it is trained on, making it hard to use. DeepDrawing was only tested on graphs with about 100 nodes. SmartGD [44] optimizes for different (combinations of) quality metrics just like SGD^2 [3], but is untested on larger graphs. Graph Neural Drawers (GND) [39] uses graph neural networks and can handle graphs of about 10K nodes. GND has a time complexity of $O(|E|^{\frac{3}{2}})$. CoReGD [23] also uses graph neural networks and can handle graphs of roughly 25K nodes. Yet, its complexity of $O(N \log(N))$ is above our linear target.

2.2 Dimensionality Reduction

Given a dataset $\mathbf{X} = \{\mathbf{x}_i\} \subset \mathbb{R}^n$, dimensionality reduction (DR) methods, also called projections, $M : \mathbf{X} \to \mathbf{Y}$, create a dataset $\mathbf{Y} = \{\mathbf{y}_i\} \subset R^m$, $m \ll n$ which preserves the so-called *data structure* of $\mathbf{X}$, such that points close (resp. far) in R^n are mapped to points close (resp. far) in R^m. DR methods accept as input either $\mathbf{X}$ or a matrix of the pairwise distances between its data points. Many DR methods exist, each of which preserves different data structure aspects and, as such, produce different drawings of the same dataset, differing significantly in terms of quality, speed, robustness, out-of-sample ability, and ease of use [15].

NNP (Neural Network Projection) [14] is a meta-approach that aims to solve the scalability and lack of out-of-sample ability for any other projection technique. Given a subset of $\mathbf{X}' \subset \mathbf{X}$ (a few thousand points) and its projection $\mathbf{X}'$ computed by any user-chosen method, NNP learns the mapping $X' \to M(\mathbf{X}')$ by a simple neural network (3 fully connected hidden layers, sizes 256, 512, 256). This mapping is used to project the entire dataset $\mathbf{X}$. NNP creates projections of quality close to the ground truth in time $O(|\mathbf{X}|)$; is robust to small changes in the input data [6]; can imitate any user-chosen projection; handles datasets of any dimensionality n; and has no free user set parameters. As such, NNP is an ideal candidate

to replace t-SNE in tsNET for graph drawing. Yet, a key obstacle exists: NNP needs a high-dimensional *dataset* $\mathbf{X}$ as input, not a distance *matrix* computed on $\mathbf{X}$, making it not directly applicable to graphs. We describe next how we overcome this limitation.

3 Method

As outlined earlier, we aim to replace the t-SNE step of tsNET with NNP. For this, we must solve two problems: Given a graph G, how to (1) reduce G to a high-dimensional dataset, as needed by NNP; (2) extract a representative subset of G to train NNP on. We address both these problems next. An overview of the NNP-NET pipeline is shown in Fig. 1. Psuedo code for the pipeline can be found in Appendix E.

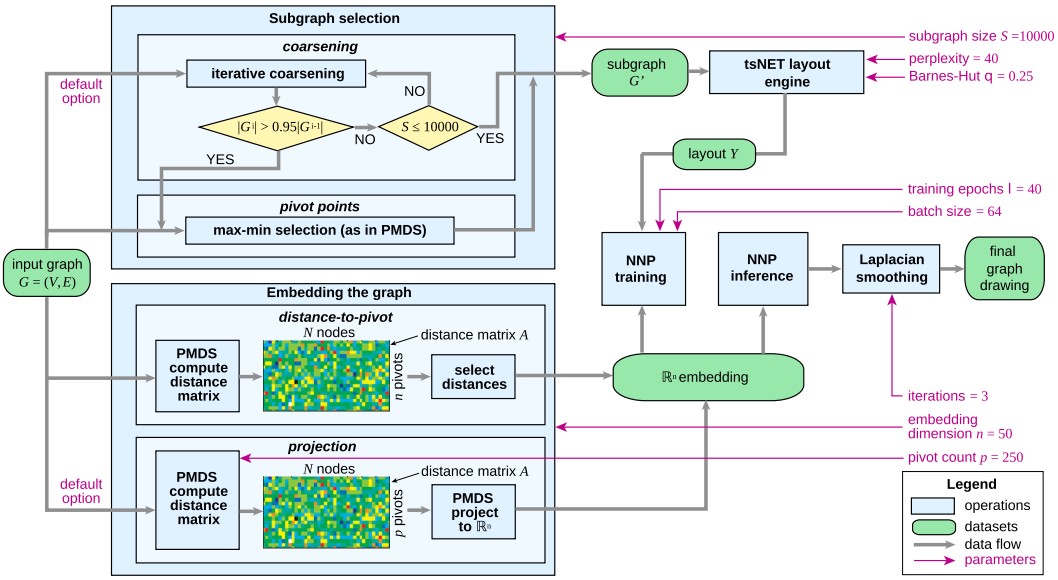


Figure 1 NNP-NET pipeline. For details, see Sec. 3.

3.1 Mapping the Graph to High-dimensional Data

NNP requires a dataset $\mathbf{X} = \{x_i\} \subset \mathbb{R}^n$ as input, where n is a free-to-choose parameter. Creating $\mathbf{X}$ from G in linear time in relation to the graph size, one of our key goals, rules out using G's full distance matrix. We propose two methods to create the embedding $\mathbf{E} = \{e(v)\}_{v \in V}$: (1) *distance-to-pivot*, uses a smaller matrix A as $\mathbf{E}$, computed the same way as PMDS [4] computes its pivot matrix; and (2) *projection*, project v to $\mathbb{R}^n$ dimensions via PMDS using $p = 250$ pivots. Method (2) was chosen for its better layout quality while sacrificing speed (see Appendix A). Another advantage of (2) is that PMDS can be freely swapped with any other projection technique. Note also that, while PMDS cannot project data to 2D accurately in general, we only use it as an *intermediate* mapping (to $n \gg 2$ dimensions); the final 2D layout is created by NNP. Setting $n = 50$ balances well between quality and computing time (see Appendix B).

3.2 Extracting Suitable Training Data

To train NNP, we need to extract a small but representative subgraph $G' \subset G$. Following various tests (Appendix C), we set the number of nodes in our subgraph of our training set to be equal to $S = 10000$. Then $G' = (V', E')$ and $S = |V'|$. Many multilevel GD methods extract such a subgraph G', *e.g.*, SFDP [27] and FM^3 [22]. These multilevel layout techniques differ in that (1) they reduce G until it is not useful to reduce any further, while we reduce to a *fixed* size S; (2) they care about all intermediate levels; we only care about the final G'.

To consistently reduce G to S nodes, we use a multilevel scheme similar to DRGraph [47], *i.e.*, *coarsen* G into a sequence $G, G^1, G^2, \ldots, G'$ of increasingly smaller graphs. Each iteration, we repeatedly choose a center node $\mathbf{c} \in G^i$ from all not yet clustered nodes that has the lowest neighbor count. All not-yet-clustered direct neighbors of $\mathbf{c}$ are added to $\mathbf{c}$'s cluster. The iteration ends when all nodes of G^i are clustered. Finally, we create G^{i+1} using the cluster centers $\mathbf{c}$ as nodes, and connecting nodes corresponding to adjacent clusters. Iterations continue until we reach the target size S.

Not all graphs can be reduced to the target size S by this coarsening method. To handle this, we use a stopping heuristic that exits coarsening when $|G^i| > 0.95|G^{i-1}|$. We then use a slower backup method to create G' by selecting S such points from G^i. This is done using *pivot-points* in a max-min approach, like PMDS [4]– choose a first node randomly, then add nodes in order of highest minimum distance to all already chosen nodes until we get S nodes. This gives good results but has a complexity of $O(NS)$. Appendix D compares the pivot-points and coarsening methods showing that the latter yields better quality-*vs*-speed.

After constructing the subgraph G', we lay out G' using tsNET* and train NNP to mimic the produced layout when given its corresponding embedding.

3.3 Smoothing

Training NNP on the subgraph G' gives good results in terms of training error (see next Sec. 4). Yet, the produced layouts contain some amount of high-frequency noise, visible in *e.g.* graphs with a grid-like structure (Fig. 2 left). We remove such noise using three Laplacian smoothing iterations (Fig. 2 right). Weights were used as follows:

$$v_i = \frac{1}{\sum_{j \in V_i} 1/w_{ij}} \sum_{j \in V_i} \frac{v_j}{w_{ij}}, \tag{1}$$

where V_i are all direct neighbors of v_i. This is done for all nodes in the graph.

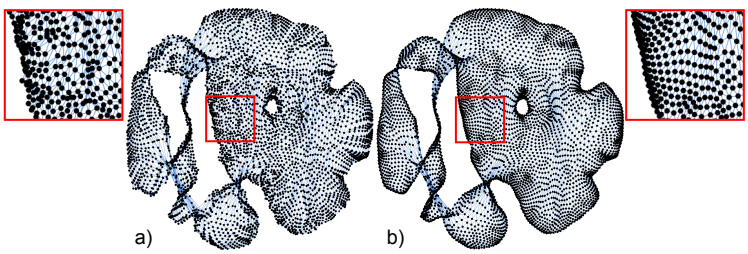


Figure 2 (a) NNP-NET drawing of the 3elt graph. (b) Effect of 2 Laplacian smoothing passes.

3.4 Complexity

Table 1 shows the time complexity of all steps of our method. For creating the $\mathbb{R}^n$ embedding via PMDS (Sec. 3.1), both the dimension count n and pivot count p are constants, yielding linear complexity in the number of nodes N. For creating the subgraph G' (Sec. 3.2), if our coarsening successfully reaches the target size S, the cost reduces to $O(iN)$; the factor $|G'|N$ accounts for using the pivot fallback. Moreover, when using coarsening, since G^k is reduced by at least a factor of 5% at each coarsening step, i is upper bounded by $\sum_{k=0}^{\infty}(0.95)^k = 20$, so the cost term $O(iN)$ is indeed linear in N. Generating ground truth by running ts-NET* on G' does not depend on the input size N. Training also does not depend on N and is linear in the number of training epochs λ, which is set to a constant value. Inference is linear in N. Altogether, the end-to-end time complexity of NNP-NET is linear in its input size N. Separately, we note that NNP-NET's space complexity is also linear in N.

Table 1 Time complexity of all steps of NNP-NET.

Algorithm step	Time complexity	Parameters
Create the $\mathbb{R}^n$ embedding of G(Sec. 3.1)	$O(npN + p^2N)$	n dimensions, p pivots
Create subgraph G' of size S (Sec. 3.2)	$O(iN + SN)$	i iterations, S nodes
Create ground truth by running ts-NET* on G'	$O(S\log(S\|))$	none
NNP training	$O(\lambda nS)$	λ training epochs
NNP inference to lay out G	$O(nN)$	none
Laplacian smoothing (Sec. 3.3)	$O(N)$	none

4 Results

4.1 Experimental Setup

Implementation. We implement NNP-NET in C++ (code publicly available [26]). We use PMDS from OGDF [8], modified to allow for $n > 3$ output dimensions. We implement tsNET(*) [30] to use both exact t-SNE and the tree-based Barnes-Hut t-SNE approximation [41] which reduces projection cost to $O(N\log(N))$ from $O(N^2)$; we also parallelized tsNET(*) to use multiple CPU cores. We implement NNP with TensorFlow [1] using the CPU as the GPU proved slower in our tests due to the small size of this neural network.

Datasets and techniques. Table 2 (left) shows the graphs used in our testing, all coming from the SparseSuite collection [9]. Most graphs used are on the larger side since a key goal we have is to speed up tsNET (Sec. 1). We compare NNP-NET with SFDP [27], FM^3 [22], PMDS [4], DRGraph [47], all state-of-the-art GD methods that aim to handle large graphs; and with tsNET [30] given we aim to mimic this method with reduced execution time.

Parameter values. FM^3 uses the OGDF implementation with default parameters. SFDP uses the GraphViz implementation [13]. For DRGraph [47], we use the authors' implementation with their suggested parameter values. Tab. 2 (right) lists all NNP-NET parameter values. We compute the tsNET* ground truth on the full graph G instead of G' when $S > N$. All tests are run on a PC with an Intel i7-11370H CPU and 16 GB of RAM.

Table 2 Left: Graph datasets used in the evaluation. All graphs come from SparseSuite [9]. Right: Parameters used for NNP-NET.

Graph dataset	$\|V\|$	$\|E\|$	Weights
dwt_1005	1005	3808	
sierpinski3d	2050	6144	
MISKnowledgeMap	2427	28511	✓
3elt	4720	13722	
optdigits_10NN	5620	39825	✓
fe_4elt2	11143	32818	
bcsstk36	23052	1143140	
k49_norm_10NN	38547	309079	✓
fe_bcsstk32	44609	985046	
m_t1	97578	9753570	
ship_003	121728	3777036	
fe_ocean	143437	819186	
ok2010	269118	1274148	✓
web-NotreDame	325729	1497134	
coPapersCiteseer	434102	16036720	
gsm_106857	589446	21758924	
tx2010	914231	4456272	✓
com-Youtube	1134890	5975248	
Flan_1565	1564794	59485419	
com-LiveJournal	3997962	34681189	

Parameter	Value
subgraph size $S = \|G'\|$	10000
n (embedding size)	50
Laplacian smoothing passes	3
t-SNE perplexity	40
θ (Barnes-Hut approximation)	0.25
p (PMDS pivot points)	250
batch size (NNP training)	64
epochs λ (NNP training)	40

4.2 Quality Metrics

We assess the quality of graph drawings using neighborhood preservation, stress, and Shepard diagrams, in line with [30, 47]. Metrics like edge length deviation [29], crossing number [37], and node-edge occlusion [12], though frequently used in GD literature, are not very informative for very large graphs.

Neighborhood preservation measures how well neighborhoods in G are preserved in the drawing $\mathbf{Y}$ [36]. Let $N_G(v_i, r_G) = \{v_j \in V | d_{ij} \leq r_G\}$ be the set of nodes v_j with a *graph-theoretic distance* d_{ij} of at most r_G from v_i; we set $r_G = 2$ following [30]. Let $N_Y(\mathbf{p}_i, k_i)$ be the set of nodes whose drawings $\mathbf{p}_j$ are the k_i-nearest-neighbors of $\mathbf{p}_i$ in the 2D layout space, where $k_i = |N_G(y_i, r_G)|$. Neighborhood preservation

$$\nu = \frac{1}{|\mathbf{V}|} \sum_i \frac{|N_G(v_i, r_G) \cap N_Y(\mathbf{p}_i, k_i)|}{|N_G(v_i, r_G) \cup N_Y(\mathbf{p}_i, k_i)|} \in [0, 1] \tag{2}$$

measures how similar $N_G(v_i, r_G)$ and $N_Y(\mathbf{p}_i, k_i)$ are (higher values are better).

Normalized stress measures how well the graph-theoretical distances d_{ij} between all node pairs are preserved by Euclidean distances in the graph drawing by

$$\sigma = \min_a \frac{1}{|V|^2} \sum_{i \neq j} \frac{(d_{ij} - a\|\mathbf{p}_i - \mathbf{p}_j\|)^2}{{d_{ij}}^2} \in [0, 1], \tag{3}$$

with lower values indicating better distance preservation. The factor a scales the drawing in order to minimize stress for a fair comparison. Following [23], we compute it using

$$a = \frac{\sum_{i \neq j} \|\mathbf{p}_i - \mathbf{p}_j\| / d_{ij}}{\sum_{i \neq j} \|\mathbf{p}_i - \mathbf{p}_j\|^2 / d_{ij}^2}. \tag{4}$$

Table 3 shows neighborhood preservation ν for all our tested drawings. Results for the largest graph *com-LiveJournal* are missing as it took too long to evaluate Eqn. 2. Other missing values tell that the respective GD method could not complete on the respective graph due to running out of RAM or took longer than 2 hours to compute. The NNP-NET results are very close to tsNET* for most graphs, with NNP-NET doing worse for *e.g. MISKnowledgeMap* and *k49_norm_10NN* but much better for *fe_bcsstk32*. NNP-NET results are consistently better than PMDS except for *com_YouTube*. When compared with DRGraph, results are more mixed, with no clear winner. Table 4 shows stress σ for all generated drawings. NNP-NET yields very similar values to tsNET* (both versions). Similarly to ν, we do not see a clear winner between NNP-NET and DRGraph. PMDS, SFDP and FM^3 yield lower σ values than NNP-NET – not surprising since these methods optimize for stress while NNP-NET (like t-SNE) optimizes for neighborhood preservation.

Table 3 Neighborhood preservation ν per graph and layout method (higher is better).

	Neighborhood preservation per graph (higher is better)						
Graph	SFDP	FM^3	PMDS	DRGraph	tsNET* exact	tsNET* approx	NNP-NET
dwt_1005	0.5	0.53	0.47	0.5	0.62	0.59	0.53
sierpinski3d	0.51	0.51	0.2	0.52	0.55	0.53	0.52
MISKnowledgeMap	0.17	0.16	0.13	0.33	0.4	0.41	0.24
3elt	0.62	0.65	0.36	0.63	0.66	0.63	0.63
optdigits_10NN	0.52	0.5	0.35	0.62	0.61	0.63	0.61
fe_4elt2	0.47	0.52	0.25	0.56	0.6	0.59	0.59
bcsstk36	0.45	0.41	0.3	0.49	-	0.51	0.44
k49_norm_10NN	0.048	0.045	0.034	0.12	-	0.18	0.093
fe_bcsstk32	0.32	0.38	0.21	0.41	-	0.24	0.37
m_t1	0.3	0.34	0.21	0.37	-	0.35	0.32
ship_003	0.21	0.21	0.17	0.24	-	0.2	0.24
fe_ocean	0.11	0.12	0.09	0.12	-	-	0.09
ok2010	0.48	0.46	0.27	0.32	-	-	0.44
web-NotreDame	0.38	0.31	0.33	0.46	-	-	0.39
coPapersCiteseer	-	0.079	0.058	0.17	-	-	0.091
gsm_106857	-	0.17	0.12	0.24	-	-	0.21
tx2010	0.42	0.39	0.26	0.21	-	-	0.36
com-Youtube	0.015	-	0.092	0.055	-	-	0.069
Flan_1565	-	-	0.09	0.2	-	-	0.21

Shepard diagrams refine distance-preservation insights captured by stress. These diagrams are scatterplots that show how distances between point-pairs correlate over two different spaces [5, 17, 47], with graph distance on the x-axis. Figure 3 shows these diagrams for our tested graphs and methods. Scatterplots close to the main diagonal indicate cases where the layout preserves graph distances well, *e.g.*, for *dwt_1005* and *ship_003*, for all methods. Plots falling largely under this diagonal indicate layouts where the layout cannot "unclutter" the graph. We see how this occurs for some of the very large graphs *e.g. com-YouTube* and *com-LiveJournal*, unsurprising, given the difficulty to lay out such complex structures. Plots falling above the diagonal indicate layouts where points are placed too far away from each other, or, in other words, too long edges drawn in the layout, see *e.g. fe_bccstk32* (ts-NET* approx). Overall, we see that SFDP, FM^3, and PMDS preserve distances better than the other methods, which is expected given their design. The remaining methods – which all are based on ts-NET, so favor neighborhood preservation, have quite similar patterns of distance preservation on the smaller graphs (Fig. 3 left column). For the larger graphs, which only DRGraph and NNP-NET can handle, we observe either similar patterns or a tendency for DRGraph to underestimate 2D distances more than NNP-NET, *i.e.*, clutter nodes – see *e.g. tx_2010*, *CoPapers-CiteSeer*, and *ok_2010*. Overall, we conclude that NNP-NET performs at least as well as the other t-SNE based methods with respect to distance preservation.

Table 4 Stress σ per graph and method. Lower is better.

	Stress per graph (lower is better)						
Graph	SFDP	FM^3	PMDS	DRGraph	tsNET* *exact*	tsNET* *approx*	NNP-NET
dwt_1005	0.029	0.026	0.029	0.03	0.038	0.036	0.029
sierpinski3d	0.079	0.078	0.104	0.08	0.077	0.086	0.125
MISKnowledgeMap	0.149	0.171	0.159	0.191	0.188	0.186	0.199
3elt	0.057	0.062	0.056	0.059	0.08	0.079	0.073
optdigits_10NN	0.159	0.144	0.125	0.172	0.142	0.14	0.139
fe_4elt2	0.05	0.069	0.064	0.068	0.095	0.098	0.097
bcsstk36	0.069	0.071	0.068	0.074	-	0.11	0.088
k49_norm_10NN	0.162	0.153	0.155	0.173	-	0.168	0.165
fe_bcsstk32	0.138	0.093	0.098	0.175	-	0.187	0.179
m_t1	0.095	0.079	0.074	0.103	-	0.078	0.147
ship_003	0.069	0.086	0.036	0.042	-	0.059	0.079
fe_ocean	0.04	0.036	0.045	0.106	-	-	0.3
ok2010	0.116	0.123	0.084	0.316	-	-	0.121
web-NotreDame	0.226	0.178	0.319	0.492	-	-	0.314
coPapersCiteseer	-	0.185	0.249	0.245	-	-	0.215
gsm_106857	-	0.089	0.102	0.131	-	-	0.106
tx2010	0.13	0.159	0.058	0.705	-	-	0.137
com-Youtube	0.179	-	0.275	0.367	-	-	0.427
Flan_1565	-	-	0.093	0.093	-	-	0.089
com-LiveJournal	-	-	0.206	0.305	-	-	0.27

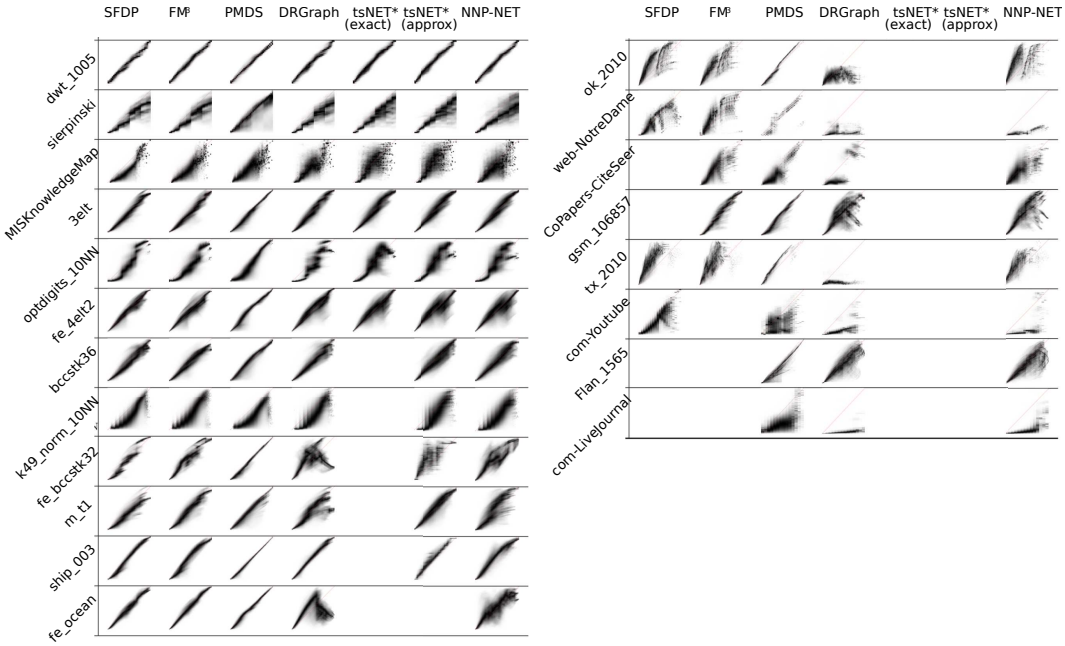


Figure 3 Shepard distance-preservation diagrams for all tested methods on all graphs.

4.3 Visual Comparison

Table 5 shows the drawings created by all tested methods on all graphs, with edges drawn half-transparent to limit visual clutter. As in Tables 3 and 4, missing entries indicate methods that fail due to memory constraints or took too long to complete. We see that exact tsNET* cannot handle graphs larger than *fe_4elt2* (11143 nodes) due to its high runtime cost. Approximate ts-NET* scales a bit better given its faster Barnes-Hut t-SNE implementation but hits a limit beyond *ship_003* (121728 nodes). NNP-NET can handle all graphs. For the

smaller graphs, NNP-NET yields drawings that are visually very similar to the ground-truth ones from approximate tsNET*. Interestingly, for *fe_bcsstk32* and *ship_003*, approximate tsNET* shows unexpected behavior where all nodes snap to a grid structure. NNP-NET does not copy this behavior but yields more plausible drawings. For larger graphs (*bcsstk36* and larger), NNP-NET creates different drawings from the other tested methods that can handle such dataset sizes, *i.e.*, SFDP, FM^3, PMDS, and DRGraph. NNP-NET spreads the drawing better over the 2D space. We argue that this is desirable as it allows one to better disentangle the depicted structures.

4.4 Drawing Weighted Graphs

NNP-NET uses the edge weights of the graph if they are provided. To show their effect, we ran NNP-NET on a set of weighted graphs with and without using the weights (Fig. 4). We see that weights affect indeed the obtained drawings (as they should), most visibly for *ok2010* and *tx2010*, where the unweighted drawings show more clutter and the weighted ones a more structured, network-of-corridors-like, one. Table 6 shows that both stress and neighborhood preservation improved when including weights.

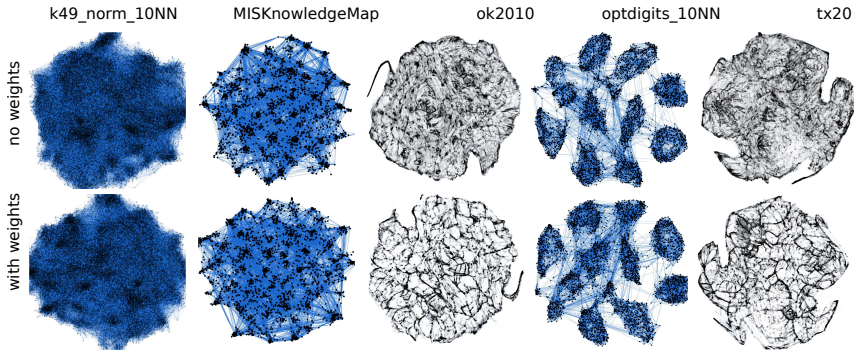


Figure 4 Comparison of NNP-NET drawings, with and without considering weights (Sec. 4.4).

4.5 Execution Time

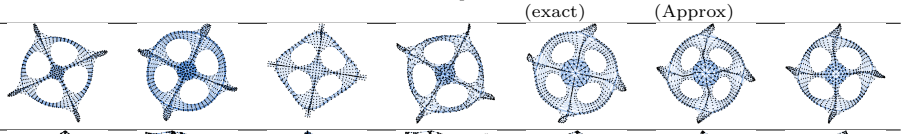
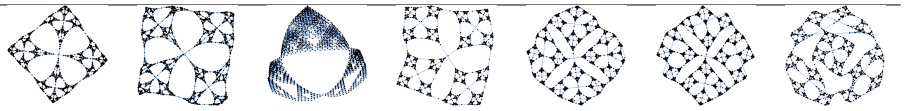
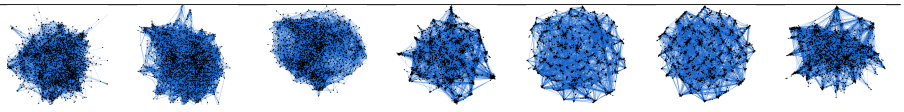
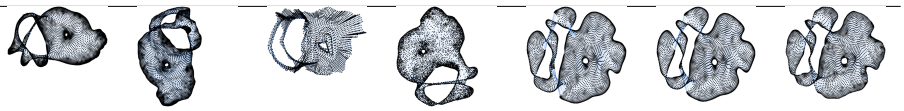
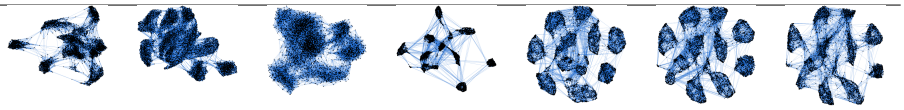
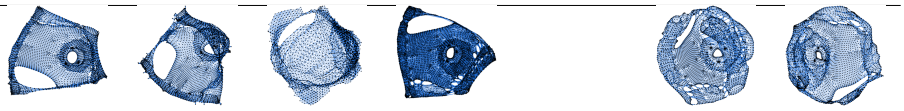
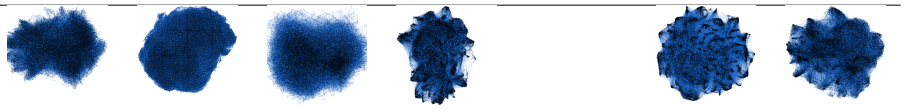
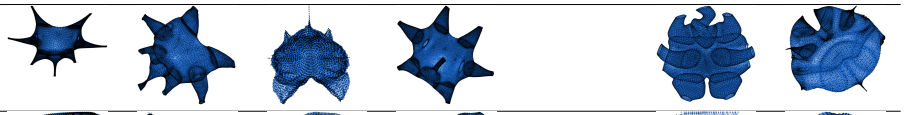
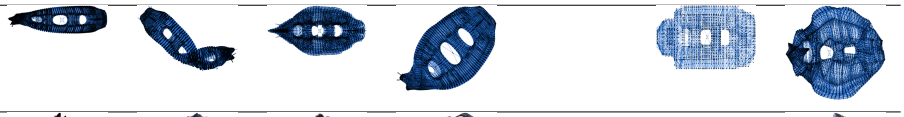
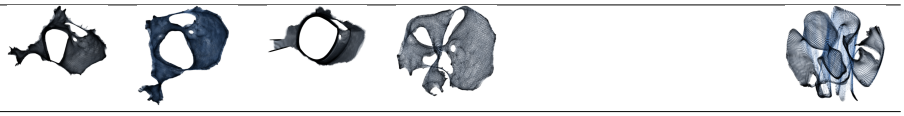
Table 7 shows the execution times for all methods tested with missing values as already explained. NNP-NET takes significantly longer on smaller graphs due to the dominating high constant cost of computing the ground truth. For larger graphs, NNP-NET is slower than PMDS, but yields better quality (see Tabs. 4 and 3). Compared to DRGraph, we see a pattern reversal, with NNP-NET becoming faster for *Flan_1565* and *com-LiveJournal*. To better examine scalability, Fig. 5 shows the time each step of NNP-NET takes *vs* the node count N (both axes use a logarithmic scale). The following trends are visible:

Embedding creation is roughly linear with N. The spiking outliers in the plot correspond to *weighted* graphs which require Dijkstra's algorithm instead of the much faster Breadth-First Search (BFS) to compute shortest paths.

Subgraph creation follows the same linear trend in N. The three outliers indicate graphs where our coarsening method could not reach the desired node count $S = 10^4$ and had to use the much slower pivot method (Sec. 3.2).

Ground truth creation is roughly constant for $N > 10^4$ nodes, a value matching S.

Table 5 Drawings created by various methods on various graphs (continued on next page).

Graph	SFDP	FM^3	PMDS	DRGraph	tsNET* (exact)	tsNET* (Approx)	NNP-NET
dwt_1005							
sierpinski3d							
MIS-Knowledge-Map							
3elt							
optdigits_10NN							
fe_4elt2							
bcsstk36							
k49_norm_10NN							
fe_bcsstk32							
m_t1							
ship_003							
fe_ocean							

Training and inference are roughly linear up to about 10^4 nodes, mainly due to the training cost. After this point, training time becomes constant. The low curve slope for larger N values tells that inference costs relatively very little and scales very well with N.

Table 5 Drawings created by various methods on various graphs (continued from previous page).

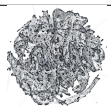
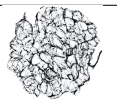
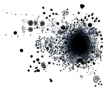
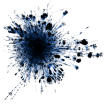
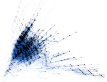
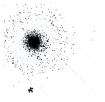
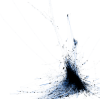
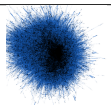
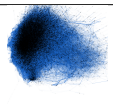
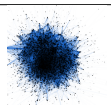
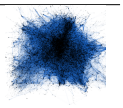
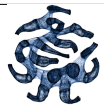
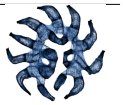
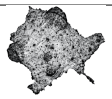
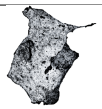
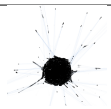
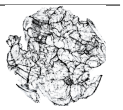
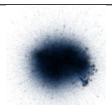
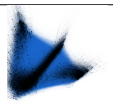
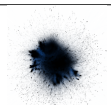
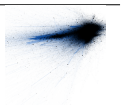
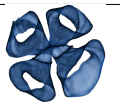
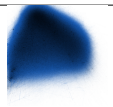
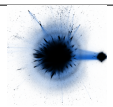
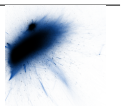
Graph	SFDP	FM^3	PMDS	DRGraph	tsNET* (exact)	tsNET* (Approx)	NNP-NET
ok2010							
web-NotreDame							
CoPapers-Citeseer							
gsm_106857							
tx2010							
com-Youtube							
Flan_1565							
com-LiveJournal							

Table 6 Quality metrics for NNP-NET drawing when using *vs* not using edge weights (Sec. 4.4).

Graph	Stress σ		Neighborhood preservation ν	
	weights	no weights	weights	no weights
k49_norm_10NN	**0.171**	0.172	**0.081**	0.077
MISKnowledge-Map	0.185	**0.183**	**0.376**	0.364
ok2010	**0.118**	0.122	**0.433**	0.382
optdigits_10NN	**0.139**	0.143	**0.613**	0.611
tx2010	**0.138**	0.144	**0.340**	0.322

We further compare NNP-NET's scalability with DRGraph, its strongest competitor *vs* ability to handle very large graphs with good quality values. For this, we took the following steps. (1) We ran NNP-NET using BFS for the weighted graphs when calculating graph distances, *i.e.*, ignoring weights – a fair comparison since DRGraph does not use weights. (2) When using the slow pivot method for subgraph creation, we accounted a time of 1 second, equal to the largest cost for all cases where this slow fallback was not needed. This is to simulate a situation where the fallback is not needed.

Figure 6 compares the cost of DRGraph (blue) with NNP-NET ran normally (orange) and with the theoretical cost of NNP-NET where we would not need the slow pivot method (steps 1 and 2 above; green). DRGraph shows a linear time complexity. In contrast, NNP-NET

Table 7 Execution time in seconds for all tested methods, all graphs (Sec. 4.5).

	Execution time per method						
Graph	SFDP	FM^3	PMDS	DRGraph	tsNET* *exact*	tsNET* *approx*	NNP-NET
dwt_1005	0.34	0.12	**0.02**	0.06	1.44	2.61	10.13
sierpinski3d	0.75	0.12	1.37	**0.11**	6.52	5.42	11.38
MISKnowledgeMap	1.23	0.76	1.20	**0.13**	40.42	13.71	24.27
3elt	1.71	0.20	**0.07**	0.27	110.70	18.92	21.14
optdigits_10NN	2.59	0.58	2.38	**0.30**	155.46	59.68	36.88
fe_4elt2	6.06	0.22	**0.15**	0.59	931.18	107.97	58.96
bcsstk36	10.89	2.28	**0.53**	1.49	-	288.91	56.96
k49_norm_10NN	37.28	**2.14**	37.13	2.66	-	1718.79	156.79
fe_bcsstk32	28.25	3.07	**2.47**	3.64	-	572.96	86.68
m_t1	77.61	14.22	**4.04**	8.21	-	1133.24	87.81
ship_003	94.44	14.24	**5.85**	9.61	-	1456.06	66.40
fe_ocean	117.74	**1.80**	2.28	14.39	-	-	94.24
ok2010	260.64	**4.13**	44.83	23.98	-	-	94.18
web-NotreDame	270.48	10.26	**6.13**	25.49	-	-	166.56
coPapersCiteseer	-	99.34	**23.69**	45.49	-	-	98.20
gsm_106857	-	57.28	**35.35**	55.86	-	-	160.41
tx2010	1165.55	**12.95**	200.21	83.18	-	-	254.17
com-Youtube	2209.56	-	**23.47**	132.47	-	-	1293.10
Flan_1565	-	-	**55.92**	171.44	-	-	124.12
com-LiveJournal	-	-	**220.14**	618.30	-	-	394.85

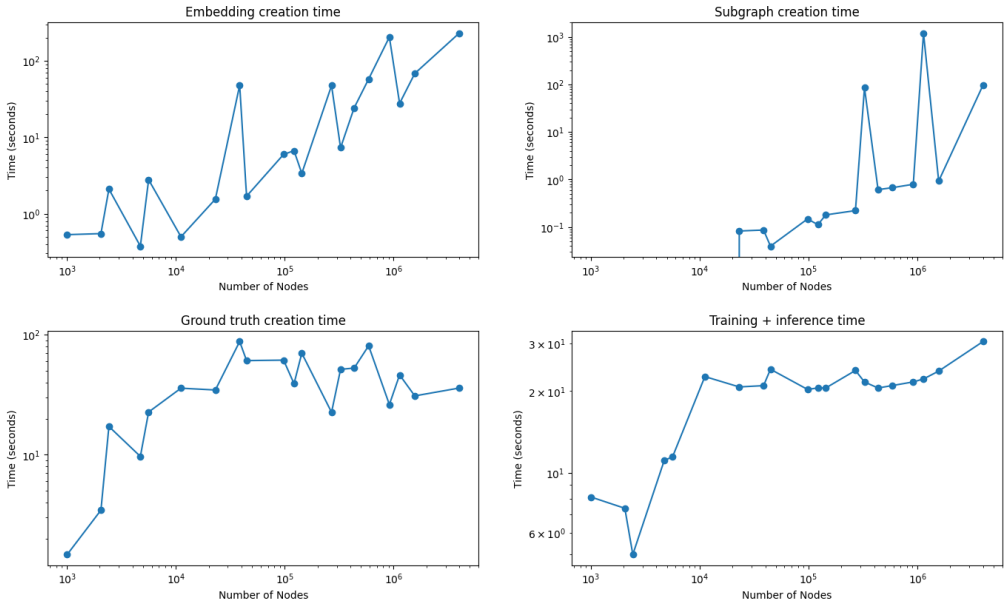


Figure 5 Execution time of each individual component of NNP-NET (Sec. 4.5).

starts with a much higher execution time. For over roughly $N = 10^4$ nodes, its cost appears almost constant (green curve) if we eliminate the effects due to (1) and (2). Actual timings show the same trend, albeit with a bit more noise (orange curve). As Tab. 7 also showed, we see how NNP-NET overtakes DRGraph at about $N = 10^6$ nodes. Given NNP-NET's and DRGraph's appearent trend, NNP-NET will likely stay faster compared to DRGraph for larger graphs.

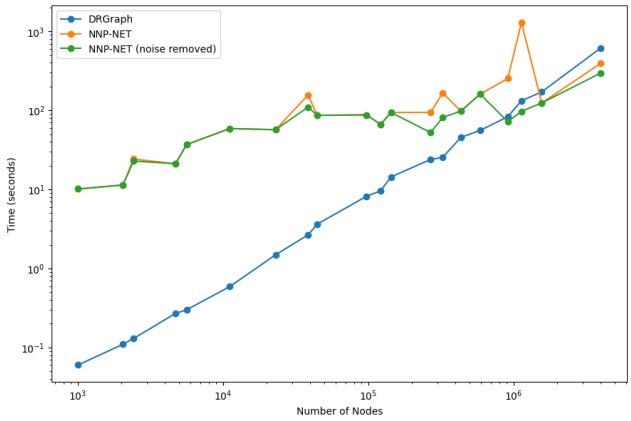


Figure 6 Execution time comparison between DRGraph, NNP-NET, and a theoretical execution time of NNP-NET with reduced noise as explained in sec. 4.5.

5 Discussion

We next discuss how NNP-NET fulfills the contributions outlined earlier in Sec. 1.

Scalability: NNP-NET has linear time complexity in the graph node count N. In practice, it takes significantly more time than other linear time complexity GD methods for smaller graphs, becoming more competitive for larger graphs, eventually outperforming DRGraph.

Layout quality: The layouts generated by our method should be of similar quality to the original tsNET* results. On smaller graphs, where we use the entire graph as ground truth, our results are very close to the approximate tsNET* results. Layouts generated for larger graphs using NNP-NET visually look good when comparing with competing methods.

Robustness: NNP-NET could handle all the tested graphs with good results. No component in NNP-NET's pipeline uses complex (parameter-dependent) heuristics. As such, we can reasonably claim that our method should be able to handle any graph dataset.

Ease of use: While our method has a number parameters that could be adjusted, all of these were set to the same fixed values for all tests (Tab. 2 right), yielding consistent good results. As such, we claim our method is practically parameter-free.

Edge weights: NNP-NET directly handles weighted graphs – something that other methods such as DRGraph cannot. Including weights in the layout computation – apart from the fact that some use-cases require this – has a positive effect on the layout quality (Sec. 4.4).

We next discuss the three key steps of NNP-NET and outline strengths, limitations, and potential ways to overcome the latter.

Embedding method: NNP-NET uses PMDS to create a high-dimensional embedding vector per node. We also tested using the *distance-to-pivot* method for the embedding. This method is faster (as only a subset of the work would have to be done), but gave worse quality results in our testing (Appendix A). Since creating the embedding becomes the cost bottleneck for large graphs, it is worth further studying this approach to see if it can be used to create similar quality layouts in less time. Separately, other layout techniques than PMDS could be used to generate this embedding. Exploring such alternatives is a low-hanging fruit for further decreasing execution time.

Ground truth: As one of our key goals was to accelerate tsNET*, we logically used tsNET* to create the ground truth that NNP learns from. Using layouts created by other GD methods could give better results. Reducing the size of subgraph G', used for training, clearly decreases quality (Appendix C), which can negate the benefit of using higher-quality, but slower, methods to lay out G'. Conversely, using faster, but lower quality methods to lay out G' can significantly speed up our method for smaller graphs, but can adversely affect quality. Finding the right balance here is a topic for future work.

Subgraph computation: We tested two methods to create the subgraph G' from the full graph G: using pivot points; and iteratively coarsening G. The coarsening method was chosen as it had a significantly lower execution time (for details, see Appendix D). Yet, this method likely distorts distances between nodes. The magnitude of these distortions and their impact is not tested directly. As already outlined, the coarsening method does not always reach the target node count S. While falling back to the pivot method allows NNP-NET to handle its input, this has a high time cost. Experimenting with different alternatives for the subgraph method is topic we will look into in future work.

6 Conclusion

We presented NNP-NET, a new graph drawing method that creates layouts for very large graphs in the style of tsNET. Our results are not only very similar to the original tsNET layouts, but also have a quality comparable to DRGraph, a competing method that also uses tsNET as its base. NNP-NET leverages the speed and simplicity of Neural Network Projections (NNP) to create graph layouts to achieve higher running times, and comparable quality, on graphs over 1 million nodes, than its closest competitor, DRGraph.

Since all steps around NNP can be freely replaced, future work can focus on improvements that these steps can provide: A different embedding method could give better resulting graphs or reduce execution time. Changing the ground truth method would allow for different drawing styles. Changing the subgraph extraction method is the most important aspect to look into further, as can boost speed significantly for the graphs that our current fast coarsening heuristic cannot handle.

References

1 Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. doi:10.48550/arXiv.1603.04467.

2 Reyan Ahmed, Felice De Luca, Sabin Devkota, Stephen Kobourov, and Mingwei Li. Graph drawing via gradient descent, $(GD)^2$. In *Graph Drawing*, pages 3–17. Springer, 2020. doi:10.1007/978-3-030-68766-3_1.

3 Reyan Ahmed, Felice De Luca, Sabin Devkota, Stephen Kobourov, and Mingwei Li. Multicriteria scalable graph drawing via stochastic gradient descent, $(SGD)^2$. *IEEE TVCG*, 28(6):2388–2399, 2022. doi:10.1109/TVCG.2022.3155564.

4 Ulrik Brandes and Christian Pich. Eigensolver methods for progressive multidimensional scaling of large data. In *Graph Drawing*, pages 42–53. Springer, 2007. doi:10.1007/978-3-540-70904-6_6.

5 Ulrik Brandes and Christian Pich. An experimental study on distance-based graph drawing. In *Graph Drawing*, pages 218–229. Springer, 2009. doi:10.1007/978-3-642-00219-9_21.

6 Carlo Bredius, Zonglin Tian, and Alexandru C. Telea. Visual Exploration of Neural Network Projection Stability. In *MLVis*, 2022. doi:10.2312/mlvis.20221068.

7 David M. Chan, Roshan Rao, Forrest Huang, and John F. Canny. t-SNE-CUDA: GPU-accelerated t-SNE and its applications to modern data. In *Computer Architecture and High Performance Computing*, 2018. doi:10.1109/CAHPC.2018.8645912.

8 Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W. Klau, Karsten Klein, and Petra Mutzel. The Open Graph Drawing Framework (OGDF). In R. Tamassia, editor, *Handbook of Graph Drawing and Visualization*, pages 543–569. CRC Press, 2014.

9 Timothy A. Davis and Yifan Hu. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software*, 38(1):1–25, 2011. doi:10.1145/2049662.2049663.

10 Sabin Devkota, Reyan Ahmed, Felice De Luca, Katherine E. Isaacs, and Stephen Kobourov. Stress-plus-x (spx) graph layout. In *Graph Drawing*, pages 291–304. Springer, 2019. doi:10.1007/978-3-030-35802-0_23.

11 Peter Eades. A heuristic for graph drawing. *Congressus numerantium*, 42:146–160, 1984.

12 Peter Eades, Michael E. Houle, and Richard Webber. Finding the best viewpoints for three-dimensional graph drawings. In *Graph Drawing*. Springer, 1997. doi:10.1007/3-540-63938-1_53.

13 John Ellson, Emden R. Gansner, Lefteris Koutsofios, Stephen C. North, and Gordon Woodhull. Graphviz— open source graph drawing tools. In *Graph Drawing*, pages 483–484. Springer, 2002. doi:10.1007/3-540-45848-4_57.

14 Mateus Espadoto, Nina Hirata, Tomita Sumiko, and Alexandru C. Telea. Deep learning multidimensional projections. *Information Visualization*, 19(3):247–269, 2020. doi:10.1177/1473871620909485.

15 Mateus Espadoto, Rafael M. Martins, Andreas Kerren, Nina S. T. Hirata, and Alexandru C. Telea. Toward a quantitative survey of dimension reduction techniques. *IEEE TVCG*, 27(3):2153–2173, 2019. doi:10.1109/TVCG.2019.2944182.

16 Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software – Practice and Experience*, 21(11):1129–1164, 1991. doi:10.1002/spe.4380211102.

17 Emden R. Gansner, Yifan Hu, and Stephen C. North. A maxent-stress model for graph layout. *IEEE TVCG*, 19(6):927–940, 2013. doi:10.1109/TVCG.2012.299.

18 Emden R. Gansner, Yehuda Koren, and Stephen C. North. Graph drawing by stress majorization. In *Graph Drawing*, pages 239–250. Springer, 2005. doi:10.1007/978-3-540-31843-9_25.

19 Helen Gibson, Joe Faith, and Paul Vickers. A survey of two-dimensional graph layout techniques for information visualisation. *Information Visualization*, 12(3-4):324–357, 2013. doi:10.1177/1473871612455749.

20 Loann Giovannangeli, Frederic Lalanne, David Auber, Romain Giot, and Romain Bourqui. Deep neural network for drawing networks, $(DNN)^2$. In *Graph Drawing*, pages 375–390. Springer, 2021. doi:10.1007/978-3-030-92931-2_27.

21 Loann Giovannangeli, Frederic Lalanne, David Auber, Romain Giot, and Romain Bourqui. Toward efficient deep learning for graph drawing (*dl4gd*). *IEEE TVCG*, 30(2):1516–1532, 2024. doi:10.1109/TVCG.2022.3222186.

22 Martin Gronemann. Engineering the fast-multipole-multilevel method for multicore and SIMD architectures. *Master's thesis. Technische Univ. Dortmund*, 2009.

23 Florian Grötschla, Joël Mathys, Robert Veres, and Roger Wattenhofer. CoRe-GD: A Hierarchical Framework for Scalable Graph Visualization with GNNs. In *ICLR, Vienna, Austria*, May 2024. URL: https://openreview.net/forum?id=vtyasLn4RM.

24 Kenneth M. Hall. An r-dimensional quadratic placement algorithm. *Management science*, 17(3):219–229, 1970. doi:10.1287/mnsc.17.3.219.

25 David Harel and Yehuda Koren. Graph drawing by high-dimensional embedding. *JGAA*, 8(2):195–214, 2004. doi:10.7155/jgaa.00089.

26 Ilan Hartskeerl, Tamara Mchedlidze, Simon van Wageningen, Peter Vangorp, and Alexandru Telea. NNP-NET. Software, swhId: swh:1:dir:539da3e3adda73abc8bb1af0c2a5c59fb7abef49 (visited on 2025-11-07). URL: https://github.com/IlanHartskeerl/NNP-NET, doi:10.4230/artifacts.25058.

27 Yifan Hu. Efficient, high-quality force-directed graph drawing. *Mathematica journal*, 10(1):37–71, 2005.

28 Mathieu Jacomy, Tommaso Venturini, Sebastien Heymann, and Mathieu Bastian. ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software. *PLOS ONE*, 9(6):e98679, 2014. doi:10.1371/journal.pone.0098679.

29 Tomihisa Kamada and Satoru Kawai. An algorithm for drawing general undirected graphs. *Inf Process Lett*, 31(1):7–15, 1989. doi:10.1016/0020-0190(89)90102-6.

30 Johannes F. Kruiger, Paulo E. Rauber, Rafael M. Martins, Andreas Kerren, Stephen Kobourov, and Alexandru C. Telea. Graph layouts by t-SNE. *Computer Graphics Forum*, 36(3):283–294, 2017. doi:10.1111/cgf.13187.

31 Joseph B. Kruskal and Judith B. Seery. Designing network diagrams. In *General Conference on Social Graphics*, pages 22–50, 1980.

32 Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26, 2013. URL: https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html.

33 Stephen C. North. *Drawing graphs with NEATO*, 2004. NEATO User's Manual.

34 Fernando V. Paulovich, Alessio Arleo, and Stef van den Elzen. When dimensionality reduction meets graph (drawing) theory: Introducing a common framework, challenges and opportunities. *Computer Graphics Forum*, 44(3):e70105, 2025. doi:10.1111/cgf.70105.

35 Nicola Pezzotti, Julian Thijssen, Alexander Mordvintsev, Thomas Hollt, Baldur van Lew, Boudewijn Lelieveldt, Elmar Eisemann, and Anna Vilanova. GPGPU linear complexity t-SNE optimization. *IEEE TVCG*, 26(1):1172–1181, 2020. doi:10.1109/TVCG.2019.2934307.

36 Helen C. Purchase. Metrics for graph drawing aesthetics. *Journal of Visual Languages & Computing*, 13(5):501–516, 2002. doi:10.1006/jvlc.2002.0232.

37 Helen C. Purchase, Robert F. Cohen, and Murray James. Validating graph drawing aesthetics. In *Graph Drawing*, pages 435–446. Springer, 1996. doi:10.1007/BFb0021827.

38 Roberto Tamassia. *Handbook of graph drawing and visualization*. CRC press, 2013.

39 Matteo Tiezzi, Gabriele Ciravegna, and Marco Gori. Graph neural networks for graph drawing. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4):4668–4681, 2022. doi:10.1109/TNNLS.2022.3184967.

40 Warren S. Torgerson. Multidimensional scaling: I. Theory and method. *Psychometrika*, 17(4):401–419, 1952. doi:10.1007/BF02288916.

41 Laurens van der Maaten. Accelerating t-SNE using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014. doi:10.5555/2627435.2697068.

42 Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11), 2008.

43 Xiaoqi Wang, Kevin Yen, Yifan Hu, and Han-Wei Shen. DeepGD: A deep learning framework for graph drawing using GNN. *IEEE CGA*, 41(5):32–44, 2021. doi:10.1109/MCG.2021.3093908.

44 Xiaoqi Wang, Kevin Yen, Yifan Hu, and Han-Wei Shen. SmartGD: A gan-based graph drawing framework for diverse aesthetic goals. *IEEE TVCG*, 30(8):5666–5678, 2023. doi:10.1109/TVCG.2023.3306356.

45 Yong Wang, Zhihua Jin, Qianwen Wang, Weiwei Cui, Tengfei Ma, and Huamin Qu. DeepDrawing: A deep learning approach to graph drawing. *IEEE TVCG*, 26(1):676–686, 2020. doi:10.1109/TVCG.2019.2934798.

46 Jonathan X. Zheng, Samraat Pawar, and Dan F. M. Goodman. Graph drawing by stochastic gradient descent. *IEEE TVCG*, 25(9):2738–2748, 2019. doi:10.1109/TVCG.2018.2859997.

47 Minfeng Zhu, Wei Chen, Yuanzhe Hu, Yuxuan Hou, Liangjun Liu, and Kaiyuan Zhang. DRGraph: An efficient graph layout algorithm for large-scale graphs by dimensionality reduction. *IEEE TVCG*, 27(2):1666–1676, 2020. doi:10.1109/TVCG.2020.3030447.

A Comparison of Embedding Methods

We further compare the two embedding methods described in Sec. 3.1, namely (1) using the *distances-to-pivot* method and (2) the *projecting* method. Table 8 shows that the results obtained by the two embedding methods are visually very similar, except for *sierpinski3d*, where the distances-to-pivot method is farther from the ground-truth than the projection method.

Table 9 shows the stress, neighborhood preservation, and error *vs* the ground truth (measured by Euclidean distances between corresponding nodes in the two layouts). We see that the projection method scores a consistently lower error to ground truth compared to the distance-to-pivot method. It also has slightly better quality metric values – though these are less important since the aim of the embedding step is to recreate the ground truth as accurately as possible rather than creating the highest-quality layout. However, the projection method is significantly slower than the distance-to-pivot one.

Table 8 Layouts created using the *distance-to-pivot* and *projection* embedding methods.

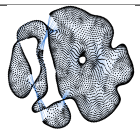
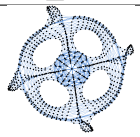
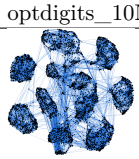
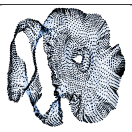
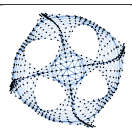
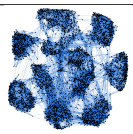
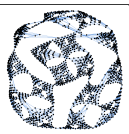
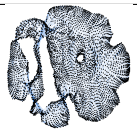
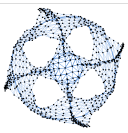
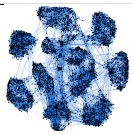
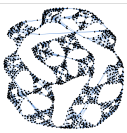
Graph	3elt	dwt_1005	optdigits_10NN	sierpinski3d
Ground truth				
Distance-to-pivot				
Projection				

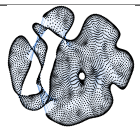
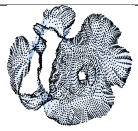
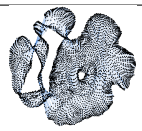
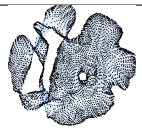
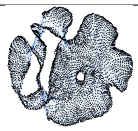
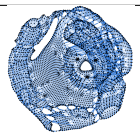
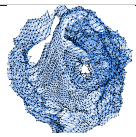
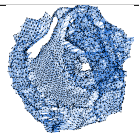
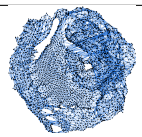
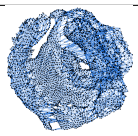
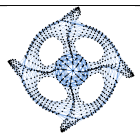
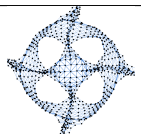
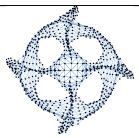
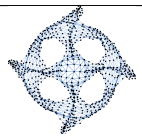
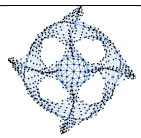
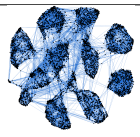
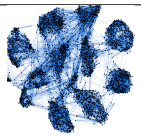
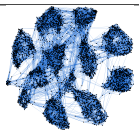
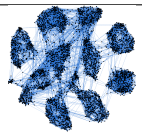
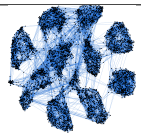
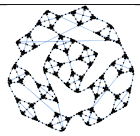
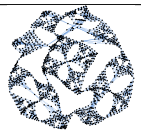
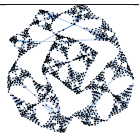
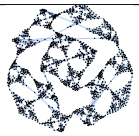
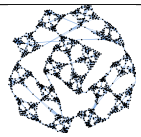
Table 9 Quality metrics obtained using the *distance-to-pivot* and *projection* embedding methods.

Graph	Embedding method	Time	Neighborhood preservation	Stress	Error *vs* ground truth
3elt	distance-to-pivot	**0.015**	0.499	0.078	0.024
3elt	projection	1.228	**0.547**	**0.077**	**0.014**
dwt_1005	distance-to-pivot	**0.007**	0.484	0.041	0.027
dwt_1005	projection	2.253	**0.521**	**0.039**	**0.019**
optdigits_10NN	distance-to-pivot	**0.799**	0.586	**0.134**	0.025
optdigits_10NN	projection	6.649	**0.598**	0.138	**0.017**
sierpinski3d	distance-to-pivot	**0.005**	0.462	**0.132**	0.028
sierpinski3d	projection	3.487	**0.498**	0.137	**0.017**

B Number of Embedding Dimensions n

We tested different values $n \in \{10, 25, 50, 100\}$ for the number of dimensions n to embed in (Sec. 3.1). Table 10 compares the obtained results and shows that these appear visually very similar for different values of n. Table 11 compares the stress, neighborhood preservation, and error to ground truth (the latter defined as in Appendix A). Neighborhood preservation and stress slightly improve with increasing n. More importantly, the error to ground truth decreases more visibly as n increases. In the same time, execution time increases quite strongly with n. Our setting $n = 50$ balances well quality and time.

Table 10 Results for different embedding sizes n.

Graph	Ground Truth	$n = 10$	$n = 25$	$n = 50$	$n = 100$
3elt					
bcsstk36					
dwt_1005					
optdigits_ 10NN					
sierpinski3d					

C Setting the Extracted Subgraph Size S

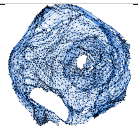
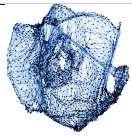
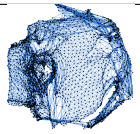
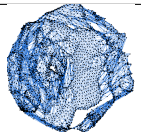
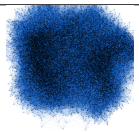
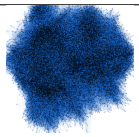
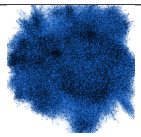
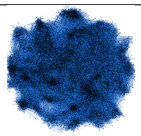
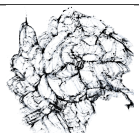
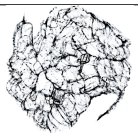
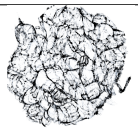
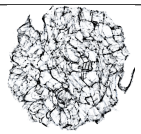
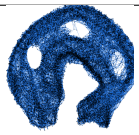
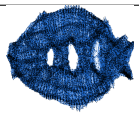
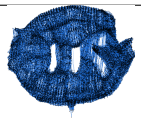
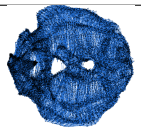
We have compared results obtained for different values $S \in \{1000, 2500, 5000, 10000\}$ for the size S of the coarsened subgraph $\mathbf{G}'$ (Sec. 3.2). Table 12 shows these results. We see that S can significantly affect the obtained layouts: for *k48_norm_10NN*, the results for $S = 10000$ show visibly more clustering than those for $S = 1000$. For *ok2010*, the results for $S = 10000$ show more individual pathways than for $S = 1000$. For *ship_003*, a bend appears for $S = 1000$, which is not present in any drawing created with higher S.

Table 13 examines the neighborhood preservation, stress, and execution time for different S values. Neighborhood preservation clearly improves with increasing S. In contrast, stress does not show a clear correlation with S. Execution time, as expected, significantly increases with S. Since, however, visual results seem to be strongly affected by S, and since we see improvement in neighborhood preservation with S, we can claim that higher S values will lead to better layouts. As such, we choose conservatively for a high default value $S = 10000$.

Table 11 Quality metrics for the different embedding sizes n.

Graph	n	Embedding Time	Neighborhood preservation	Stress	Error to GT
3elt	10	**15.5**	0.532	0.078	0.013
3elt	25	20.9	**0.580**	**0.077**	0.019
3elt	50	38.2	0.579	**0.077**	**0.011**
3elt	100	105.7	0.574	**0.077**	0.012
bcsstk36	10	**30.2**	0.411	0.102	0.027
bcsstk36	25	36.3	0.435	0.107	0.019
bcsstk36	50	54.4	0.457	**0.105**	**0.014**
bcsstk36	100	118.9	**0.463**	0.107	0.016
dwt_1005	10	**5.5**	0.504	**0.033**	0.018
dwt_1005	25	11.3	0.542	0.035	**0.012**
dwt_1005	50	28.8	0.543	0.035	**0.012**
dwt_1005	100	94.0	**0.544**	0.034	**0.012**
optdigits_10NN	10	57.7	0.607	**0.139**	0.029
optdigits_10NN	25	56.6	0.621	0.140	0.019
optdigits_10NN	50	**45.6**	0.623	0.140	0.015
optdigits_10NN	100	106.9	**0.624**	0.140	**0.014**
sierpinski3d	10	**10.7**	0.495	**0.126**	0.023
sierpinski3d	25	17.8	**0.509**	0.127	0.019
sierpinski3d	50	35.6	0.507	0.131	0.021
sierpinski3d	100	99.2	0.502	0.132	**0.014**

Table 12 Results using different sizes S for the subgraph $\mathbf{G}'$.

Graph	$\lvert\mathbf{G}'\rvert = 1000$	$\lvert\mathbf{G}'\rvert = 2500$	$\lvert\mathbf{G}'\rvert = 5000$	$\lvert\mathbf{G}'\rvert = 10000$
bcsstk36				
k49_norm_10NN				
ok2010				
ship_003				

D Comparison of Subgraph Extraction Methods

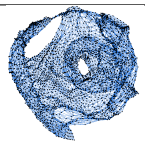
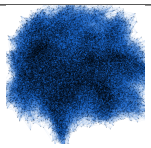
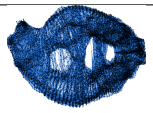
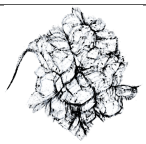
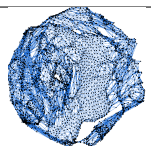
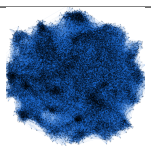
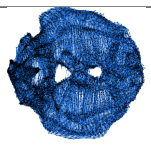
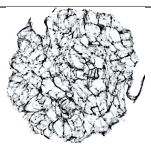
We next compare the *pivot-points* and *coarsening* methods for computing the subgraph $\mathbf{G}'$ from the input graph $\mathbf{G}$ (Sec. 3.2). Both methods receive the parameter S that determines the size of the subgraph $\mathbf{G}'$ to be extracted. Comparing results for the *same* value of S is, however, not directly useful. Indeed, the pivot-points method becomes too slow to be useful in practice for values $S > 1000$. In contrast, the coarsening method scales computationally well to higher values. As such, we chose to compare the results that both methods produce when S is set to the maximal values they accept, namely $S = 1000$ (pivot-point) and $S = 10000$ (coarsening).

Table 13 Performance metrics for the different sizes S for the subgraph $\mathbf{G}'$.

Graph	$S = \lvert\mathbf{G}'\rvert$	Neighborhood preservation	Stress	Time (seconds)
bcsstk36	1000	**0.411**	**0.085**	**32.4**
bcsstk36	2500	0.373	0.133	41.4
bcsstk36	5000	0.340	0.121	58.2
bcsstk36	10000	0.408	0.106	103.6
k49_norm_10NN	1000	0.030	0.170	**74.8**
k49_norm_10NN	2500	0.047	0.169	102.6
k49_norm_10NN	5000	0.060	**0.167**	127.1
k49_norm_10NN	10000	**0.083**	0.169	219.0
ok2010	1000	0.392	0.111	**90.2**
ok2010	2500	0.408	**0.109**	103.2
ok2010	5000	0.424	0.120	107.8
ok2010	10000	**0.436**	0.120	160.2
ship_003	1000	0.155	0.124	**45.1**
ship_003	2500	0.208	**0.073**	56.2
ship_003	5000	0.214	0.092	85.0
ship_003	10000	**0.229**	0.089	131.3

Table 14 shows the obtained results. The pivot-point method clearly yields suboptimal layouts due to its low $S = 1000$ setting – these are similar to what the coarsening method produce for $S = 1000$ (see Fig. 12). Table 15 compares the values of neighborhood preservation, stress and, execution time for the two methods. The coarsening method yields overall higher neighborhood preservation values than pivot-points (except for *bcsstk36* where it is slightly lower). Stress values are however better for pivot-points, which is not too surprising, given that those are chosen based on PMDS. Overall, the significantly higher execution time of pivot-points (for larger graphs), even when using a S value ten times smaller than for the coarsening method, determined us to choose coarsening as the default technique for subgraph extraction.

Table 14 Results obtained by using *pivot-points* and *coarsening* to extract the subgraph $\mathbf{G}'$.

Graph	bcsstk36	k49_norm_10NN	ship_003	ok2010
Pivot-points, $S = 1000$				
Coarsening, $S = 10000$				

E Pseudo Code

The following pseudo code is for the entire NNP-NET pipeline.

```
// Generate n-dimensional Embedding using PMDS
E ← PMDS(G, n)

// Generate G'
G' ← G
G^next ← ({}, {})
while size(G'.V) > S:
    V^s ← sort(G'.V) // Sorted on degree
    C ← {} // Start with no clusters
    // Choose center nodes and create their clusters
    for v ∈ V^s WHERE v ∉ C:
        C.add({v, {v_i : (v, v_i) ∈ E, v_i ∉ C}})
    for c ∈ C:
        // Add center nodes to the next iteration
        G^next.V.add(c[0])
        // Add all edges for this center node
        for c' ∈ C WHERE (v, v') ∈ G'.E : v ∈ c, v' ∈ c':
            G^next.E.add({c[0], c'[0]})
    // Check if the graph still gets reduced enough to continue
    if size(G^next.V) > size(G'.V)*0.95:
        G' ← pivotfallback(G^next, S)
    else :
        G' ← G^next
        G^next ← ({}, {})

// Generate ground truth
P^G ← tsNET*(G')
// Generating final positions
NNP ← train(E, P^G)
P ← inference(NNP, E)
P ← smoothing(P)
```

Table 15 Performance metrics using *pivot-points* and *coarsening* to extract the subgraph $\mathbf{G}'$.

Graph	Method + points	Neighborhood preservation	Stress	Time
bcsstk36	Pivot, 1000 points	**0.422**	**0.077**	**37.9**
bcsstk36	Coarsening, 10000 points	0.408	0.084	103.6
k49_norm_10NN	Pivot, 1000 points	0.045	**0.166**	**209.4**
k49_norm_10NN	Coarsening, 10000 points	**0.083**	0.171	219.0
ship_003	Pivot, 1000 points	0.194	**0.083**	**78.4**
ship_003	Coarsening, 10000 points	**0.229**	0.124	131.3
ok2010	Pivot, 1000 points	0.382	**0.087**	255.4
ok2010	Coarsening, 10000 points	**0.436**	0.112	**160.2**

The Price of Connectivity Augmentation on Planar Graphs

Hugo A. Akitaya
University of Massachusetts Lowell, MA, USA

Justin Dallant
Aarhus University, Denmark

Erik D. Demaine
CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA

Michael Kaufmann
Wilhelm-Schickard Institut für Informatik, University of Tübingen, Germany

Linda Kleist
Department of Informatics, Universität Hamburg, Germany

Frederick Stock
University of Massachusetts Lowell, MA, USA

Csaba D. Tóth
Department of Mathematics, California State University Northridge, Los Angeles, CA, USA
Department of Computer Science, Tufts University, Medford, MA, USA

Torsten Ueckerdt
Karlsruhe Institute of Technology, Germany

Abstract

Given two classes of graphs, $\mathcal{G}_1 \subseteq \mathcal{G}_2$, and a c-connected graph $G \in \mathcal{G}_1$, we wish to augment G with a smallest cardinality set of new edges F to obtain a k-connected graph $G' = (V, E \cup F) \in \mathcal{G}_2$. In general, this is the $c \to k$ connectivity augmentation problem. Previous research considered variants where $\mathcal{G}_1 = \mathcal{G}_2$ is the class of planar graphs, plane graphs, or planar straight-line graphs. In all three settings, we prove that the $c \to k$ augmentation problem is NP-complete when $2 \leq c < k \leq 5$.

However, the connectivity of the augmented graph G' is at most 5 if $\mathcal{G}_2$ is limited to planar graphs. We initiate the study of the $c \to k$ connectivity augmentation problem for arbitrary $k \in \mathbb{N}$, where $\mathcal{G}_1$ is the class of planar graphs, plane graphs, or planar straight-line graphs, and $\mathcal{G}_2$ is a beyond-planar class of graphs: ℓ-planar, ℓ-plane topological, or ℓ-plane geometric graphs. We obtain tight bounds on the tradeoffs between the desired connectivity k and the local crossing number ℓ of the augmented graph G'. We also show that our hardness results apply to this setting.

The connectivity augmentation problem for triangulations is intimately related to edge flips; and the minimum augmentation problem to the flip distance between triangulations. We prove that it is NP-complete to find the minimum flip distance between a given triangulation and a 4-connected triangulation, settling an open problem posed in 2014, and present an EPTAS for this problem.

2012 ACM Subject Classification Mathematics of computing → Paths and connectivity problems; Mathematics of computing → Graph algorithms; Theory of computation → Computational geometry

Keywords and phrases connectivity augmentation, local crossing number, flip distance

Digital Object Identifier 10.4230/LIPIcs.GD.2025.23

Related Version A full version will appear on arXiv shortly before Graph Drawing 2025.
Full Version: https://arxiv.org/abs/2509.01096

Funding *Hugo A. Akitaya*: Research supported in part by NSF grant CCF-2348067.
Csaba D. Tóth: Research supported in part by the NSF award DMS-2154347.
Torsten Ueckerdt: Research supported by the German Research Foundation DFG-520723789.

Acknowledgements The work in this paper originated at the 11th Annual Workshop on Geometry and Graphs held at the Bellairs Research Institute, 8–15 March, 2024.



© Hugo A. Akitaya, Justin Dallant, Erik D. Demaine, Michael Kaufmann, Linda Kleist, Frederick Stock, Csaba D. Tóth, and Torsten Ueckerdt;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

Connectivity augmentation is a classical problem in combinatorial optimization [21, 34]: Given a c-connected graph $G = (V, E)$ and a positive integer k, find a minimum set F of new edges such that the graph $G' = (V, E \cup F)$ is k-connected. For undirected and unweighted graphs, Jackson and Jordán [32] gave a polynomial-time algorithm for any c and k (previous results addressed $c = 2, 3, 4$ [19, 29, 55]; see also [53]). For the weighted version, finding a minimum-cost augmentation is NP-complete [22], APX-hard, and the best approximation ratio is 1.5 [51]. In the remainder of this paper, we consider unweighted undirected graphs.

We investigate connectivity augmentation over planar graphs in three different settings, Figure 1 depicts an illustrative example: In the *abstract graph setting*, G is planar, and G' must also be planar. For this variant, NP-completeness [37] and a $\frac{5}{3}$-approximation are known for $k = 2$ [20, 27]. Moreover, there exists an $O(n^3)$-time $\frac{5}{4}$-approximation algorithm for $c = 2$ and $k = 3$ [37]. In the *topological graph setting*, G and G' are plane graphs (with a fixed embedding) and G' extends the given embedding: the problem remains NP-complete for $c = 0$ and $k = 2$, but there is a linear-time algorithm for $c = 1$ and $k = 2$ [27, 28]. Finally, in the *geometric graph setting*, both G and G' are planar straight-line graphs (PSLG, for short), and G' extends the given embedding [1, 4, 6, 31, 47]. The three settings reveal deep structural properties of embedded graphs as the edges of the input graph G form topological or geometric obstructions to possible new edges.

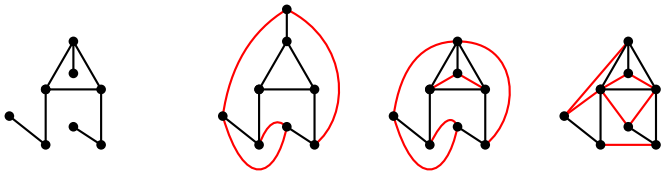


Figure 1 A planar straight-line graph, and its 3-connected minimum augmentation in the abstract, topological, and geometric settings with 4, 6, and 7 additional edges, respectively.

However, planarity severely limits the feasible values of k: Every planar graph has a vertex of degree at most 5, and hence augmentation is infeasible for any $k > 5$. Furthermore, for $n \geq 3$ points in convex position, every PSLG has a vertex of degree at most 2. The connectivity augmentation problem for PSLGs is already challenging for $k = 1, 2$; and requires additional assumptions about the point configurations for $k = 3$; see [25].

In this paper, we also consider beyond-planar variants of the problem: The augmented graph G' need not be planar but "close" to planarity. In particular, we study the trade-offs between connectivity and local crossing number, defined as follows. An abstract graph G is *ℓ-planar* if it admits a drawing in the plane such that each edge has at most ℓ crossings. The *local crossing number* of G, denoted $\mathrm{lcr}(G)$, is the minimum integer ℓ such that G is ℓ-planar.

We also consider connectivity augmentation restricted to *topological graphs* and *geometric graphs*. In both cases, V is a set of n points in the plane in general position (i.e., no three points are collinear), the edges are Jordan arcs in topological graphs and straight-line segments in geometric graphs. In both cases, the drawing of G is fixed, and the *local crossing number* is defined as the maximum number of crossings per edge *in that drawing*.

Asymptotic Bounds and Tradeoffs. We start with an asymptotically tight tradeoff between connectivity and local crossing number (regardless of the number of new edges): Every k-connected graph is $O(k^2)$-planar (Proposition 2.2). For geometric graphs on n points in

convex position, we determine the precise tradeoff (Proposition 2.3). The asymptotically tight bound can be attained even if we augment a given plane triangulation in the topological setting (Theorem 3.2). The geometric setting turns out to be much more constrained: We show that any PSLG can be augmented to a 3-connected 5-planar geometric graph, and this bound is the best possible (Theorem 8.1). However, augmenting a PSLG to a 4-connected geometric graph may already increase the local crossing number to $\Omega(n)$ (Theorem 8.3).

Algorithmic Results. Table 1 presents an overview of the computational complexity of finding minimum solutions; each field has an entry for the abstract, topological, and geometric setting (in this order); a single entry means that the same result holds in all three settings.

Table 1 Summary of complexity results. Bold results are proved in this paper. Arrows indicate that a result is implied by the one pointed to.

From/To	1	2	3	4	5
Tree	—	P [35],[27],[4]	P [35], **P**, ?	?	?
0	P	NPC [37],[28],[46]	NPC [47]	NPC [47]	NPC [47]
1	—	NPC[37], P[27], P[4]	NPC[37],[47],[4]	NPC[37],[47],[4]	NPC[37],[47],[4]
2	—	—	**NPC**	**NPC**(↓)	**NPC**(↓)
3	—	—	—	**NPC**	**NPC**(↓)
4	—	—	—	—	**NPC**

Trees. Kant [35] gave an $O(n)$-time algorithm for the 2- and 3-connectivity augmentation of outerplanar graphs with n vertices in the abstract setting, and Gutwenger et al. [27] gave an $O(n(1+\alpha(n))$-time algorithm for the $1 \to 2$ augmentation for connected plane graphs. For a PSLG tree on n vertices, there is an $O(n^4)$-time algorithm for 2-connectivity augmentation [4]. (Biconnectivity augmentation can also be solved optimally over outerplanar graphs, where both G and G' must be outerplanar [26].) We give an $O(n)$-time algorithm for the minimum augmentation of a plane tree to a 3-connected plane graph in the topological settings (Theorem 4.2). In the geometric setting, 3-connectivity augmentation of trees remains open.

Geometric Triangulations. For a straight-line triangulation on n points in convex position, we give a combinatorial characterization of 3- and 4-connected augmentations. In this setting, the dual graph is a tree. We use the dual tree for a dynamic programming algorithm to compute a minimum augmentation to a k-connected ℓ-planar geometric graph for $k \in \{3, 4\}$ and constant $\ell \in \mathbb{N}$ in $O(n)$ time (Theorem 7.2).

Hardness. In most settings, however, the $c \to k$ connectivity augmentation problem is NP-complete. We prove the following theorem using several reductions from planar 3-SAT.

▶ **Theorem 1.1.** *Given a c-connected planar graph G and an integer τ, deciding whether there is an edge set of cardinality at most τ that augments G into a planar k-connected G' is NP-complete when $c \in \{2, 3, 4\}$, $k \in \{3, 4, 5\}$, and $c < k$. The problem remains NP-complete if G is a topological or geometric graph and G' is required to be a 1-plane graph extending the embedding of G.*

Our reductions presented in Section 5 (from planar 3-SAT) work similar to [4] in the sense that the truth assignment of a variable is encoded in one of two possible perfect matchings on a subset of vertices whose degree must be augmented. The proof in [4] is specific to the

geometric setting, and it is not trivial to generalize it to the other two settings. Moreover, c-connected gadgets for $c \in \{2, 3, 4\}$ required delicate new designs to ensure that the desired behavior in the reduction [4] go through, especially when G is a triangulation.

Minimum-Degree Augmentation. In a k-connected graph G on $n > k$ vertices, the minimum vertex degree $\delta(G)$ is at least k. All new hardness reductions in Table 1 hinge on increasing the degree of a set of special vertices from c to k, using the minimum number of new edges, while maintaining planarity. As a consequence, we also prove that the following *min-degree augmentation* problems are NP-complete:

▶ **Corollary 1.2.** *Given a planar graph G with $\delta(G) = c$ and an integer k, find a minimum set F of new edges such that $G' = (V, E \cup F)$ is planar and satisfies $\delta(G') \geq k$.*

Flip Distance to a 4-Connected Triangulation. A *flip* (a.k.a. *edge exchange*) in a combinatorial triangulation (i.e., edge-maximal planar graph) is an elementary operation that removes one edge and inserts a new edge, producing another triangulation. The *flip diameter* on n vertices is the minimum number of flips that can transform any combinatorial triangulation into any other. Current bounds on the flip diameter use a canonical form, which is a triangulation with a Hamiltonian cycle. Since 4-connected triangulations are Hamiltonian [52] (see also [38, 44, 49] for extensions to beyond-planar graphs), the flip distance to a Hamiltonian or a 4-connected triangulation was studied [14, 16]. Bose et al. [13] posed the following problems: Given a combinatorial triangulation T, find the minimum number of flips that can take it to a 4-connected (resp., Hamiltonian) triangulation. Our hardness reduction for the $3 \to 4$ connectivity augmentation problem can be modified to show that finding the flip distance to 4-connectedness is also NP-complete (Corollary 1.3).

A *simultaneous flip* in a combinatorial triangulation is an operation that performs one or more edge flips simultaneously (each of which is a valid flip, and they can be performed independently) [24]. Bose et al. [12] showed that every triangulation on $n \geq 6$ vertices can be transformed into a 4-connected triangulation with a single simultaneous flip; the number of flipped edges is at most $\lfloor (2n - 7)/3 \rfloor$ [16]. We show that it is NP-complete to find a minimum simultaneous flip to 4-connectivity (Corollary 1.3). By inserting all simultaneously flipped edges (without removing any edges), we obtain a 4-connected 1-planar graph. In fact, finding a minimum augmentation to a 4-connected 1-planar graph in a triangulation is equivalent to finding a minimum simultaneous flip to 4-connectivity [12, Section 3].

▶ **Corollary 1.3.** *Given a combinatorial triangulation T and an integer τ, it is NP-complete to decide whether there exists a sequence of at most τ flips that transforms T into a 4-connected triangulation. It is also NP-complete to decide whether there is a single simultaneous flip of cardinality at most τ that transforms T into a 4-connected triangulation.*

In Section 6, we give an EPTAS for the first of the two problems (Theorem 6.1). We also give some intuition for why our techniques do not directly apply to the second problem.

Further Related Work. While we aim for increasing the *vertex-connectivity* of a given graph, analogous problems can be considered for *edge-connectivity* [5, 17, 33, 43, 50, 54]. Some of our techniques might extend to edge-connectivity, however, such adaptations are not immediate and are beyond the scope of this paper.

We have measured the distance to planarity by the local crossing number. However, there are many other notions of beyond-planarity. For example, García et al. [25] studied the connectivity augmentation problem for *geometric biplane graphs*, which admit a straight-line

drawing in the plane and a 2-coloring of the edges such that no two edges of the same color cross (a.k.a. graphs with geometric thickness 2). They showed that every sufficiently large point set in the plane in general position admits a 5-connected geometric biplane graph, and the connectivity bound 5 cannot be improved. Furthermore, every PSLG (other than a wheel or a fan) can be augmented into a 4-connected geometric biplane graph, and the connectivity bound 4 cannot be improved.

2 Tradeoff between Connectivity and Local Crossing Number

As a warmup exercise, we start with the following basic question, which corresponds to augmenting the empty graph to a k-connected graph with local crossing number at most ℓ.

▶ **Question 2.1.** *Given $\ell \in \mathbb{N}$, what is the maximum connectivity of an ℓ-planar graph?*

▶ **Proposition 2.2.** *For every $\ell \in \mathbb{N}$, the connectivity of an ℓ-planar graph is $O(\sqrt{\ell})$. This bound is the best possible and can be attained by a geometric graph on any point set in general position. Every set of $n \geq k+1$ points in the plane (in general position) admits a k-connected $O(k^2)$-planar geometric graph; and this bound is the best possible.*

Proof. Let $G = (V, E)$ be a k-connected ℓ-planar graph on $n = |V|$ vertices. Then $\deg(v) \geq k$ for every $v \in V$, and so $|E| \geq kn/2$. Assume w.l.o.g. that $k \geq 8$, hence $|E| \geq 4n$. By the Crossing Lemma [3, 15, 40], the total number of crossings in any drawing of G is $\mathrm{cr}(G) = \Omega(|E|^3/n^2) = \Omega(k^3 n)$. By the pigeonhole principle, there is an edge with $\Omega(\mathrm{cr}(G)/|E|) = \Omega(|E|^2/n^2) = \Omega(k^2)$ crossings; hence $\ell = \Omega(k^2)$ and $k = O(\sqrt{\ell})$.

For a matching lower bound, let V be the set of $n \geq k+1$ points in the plane in general position. Assume w.l.o.g. that they have distinct x-coordinates. Sort the points by x-coordinate and connect every point to its k neighbors on the left and right. The first $k+1$ points induce a complete graph, which is k-connected. By induction, the first i points also induce a k-connected graph for $k < i \leq n$. Any edge can only cross other edges with overlapping x-projections, so every edge has $O(k^2)$ crossings. ◀

Determining the precise dependency between k and ℓ remains largely open. One exception is the case $k = 1$: The maximum connectivity of a 1-planar graph is 7. Every 1-planar graph on $n \geq 3$ vertices has at most $4n - 8$ edges [45], so the minimum degree is at most 7, hence its connectivity is at most 7. This degree bound is tight: Biedl [9] constructed 1-planar graphs with minimum degree 7, which happen to be 7-connected. However, the connectivity of *optimal* 1-planar graphs (which have precisely $4n - 8$ edges) is either 4 or 6 [23, 48]; there are edge-maximal 1-planar graphs with $\frac{8}{3}n - O(1)$ edges, where the connectivity is only 2 [30].

Geometric graphs. We can determine the exact tradeoff between connectivity and local crossing number for geometric graphs when the vertices are in convex position. For integers $k, n \in \mathbb{N}$, $k < n/2$, the *k-circulant graph* $G = (V, E)$ is defined on the vertex set $V = \{v_0, \ldots, v_{n-1}\}$, where $v_i v_j \in E$ if and only if the cyclic distance between i and j is at most k, that is, $\min\{|i - j|, n - |i - j|\} \leq k$; see Figure 2.

▶ **Proposition 2.3.** *For every $k \in \mathbb{N}$ and for $n \geq 2k$ points in convex position in the plane, the k-circulant graph is $2k$-connected and is $(k^2 - k)$-planar. This is the best possible: If a geometric graph on n points in convex position has minimum degree at least $2k$, then its local crossing number is at least $k^2 - k$.*

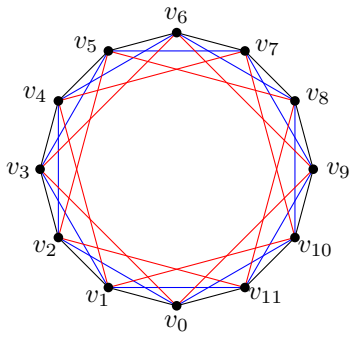


Figure 2 The 3-circulant graph on $n = 12$ vertices.

Proof. Let $k \in \mathbb{N}$, and let $G = (V, E)$ be the k-circulant graph on n vertices, that is, the vertices are on a circle, and each vertex is adjacent to k neighbors in both cw and ccw direction along the circle. Elementary counting shows that every edge crosses at most $k^2 - k$ other edges. We show that G is $2k$-connected. We need to show that $G - D$ is connected for any $D \subset V$ with $|D| = 2k - 1$. Let $s, t \in V \setminus D$. Then s and t decompose the circle into two arcs. One of them contains at most $k - 1$ vertices in D. We construct an st-path π_{st} by following this arc, and skip all vertices in D. Since we skip at most $k - 1$ consecutive vertices, all edges of π_{st} are present in $G - D$, and so $G - D$ is connected.

Now let $G = (V, E)$ be a geometric graph on $n \geq 2k$ points in convex position, with minimum degree at least $2k$. We define the *length* of an edge uv as the minimum number of edges in a path on the boundary of the convex hull. Then $|E| \geq kn$, and elementary counting shows that G has at least one edge of length at least k. Let e be a shortest edge among all edges of length at least k. We claim that if e has length $\ell \geq k$, then it has at least $k^2 - 3k + 2\ell \geq 2k^2 - k$ crossings. Assume w.l.o.g. that $V = \{v_0, \dots v_{n-1}\}$ in cyclic order along the convex hull of V, and $e = (0, \ell)$. Every edge that crosses e has on endpoint in $\{v_1, \dots, v_{\ell-1}\}$ and one in $\{v_{\ell+1}, \dots, v_{n-1}\}$. Then $\sum_{i=1}^{\ell-1} \deg(v_i) \geq 2k(\ell - 1)$. We subtract the number of vertex-edge pairs (i, e') such that $1 \leq i \leq \ell - 1\}$ and e' does not cross e. Every such edge e' has length less than k. By symmetry, it is enough to count pairs (i, e') where e' is a "left" edge, that is, $e' = \{i, j\}$ and $j < i$. The number of such edges for $i = 1, 2, \dots, \ell - 1$ is $1, 2, 3, \dots, k - 1, k - 1, \dots, k - 1$, which sums to $\binom{k}{2} + (\ell - k)(k - 1)$. So edge e crosses at least $2k(\ell - 1) - 2\left(\binom{k}{2} + (\ell - k)(k - 1)\right) = k^2 - 3k + 2\ell \geq k^2 - k$ edges. This is a lower bound for the local crossing number of G. ◀

3 Augmentation for Topological Graphs

Every edge-maximal plane graph is 3-connected, but it may have separating triangles. For augmentation to 4-connectivity, we need to go beyond planarity.

▶ **Proposition 3.1.** *Every plane graph G on $n \geq 6$ vertices can be augmented to a 4-connected 1-planar topological graph. If G is a triangulation, then $\lfloor (2n - 7)/3 \rfloor$ new edges suffice.*

Proof. Let $G = (V, E)$ be a plane graph on $n \geq 6$ vertices. We may assume that G is an edge-maximal plane graph (i.e., a triangulation). Cardinal et al. [16, Theorem 3] proved that G can be transformed into a 4-connected edge-maximal planar graph $G' = (V, E')$ using a *simultaneous flip* of at most $\lfloor (2n - 7)/3 \rfloor$ edges. Now $G = (V, E \cup E')$ is 4-connected (since it contains G') and 1-planar (an edge has a crossing iff it participates in a flip). ◀

For plane graph augmentation, we prove a tight tradeoff (matching Proposition 2.2).

▶ **Theorem 3.2.** *Every plane graph can be augmented to a k-connected $O(k^2)$-planar topological graph, and this bound is the best possible.*

Proof. Without loss of generality assume we are given a plane triangulation $G = (V, E)$ and a $k \in \mathbb{N}$. The dual graph $\mathcal{D}$ of G is a planar graph of maximum degree 3. We can partition $\mathcal{D}$ into a collection $\mathcal{C} = \{C_1, \ldots, C_t\}$ of connected subgraphs, each containing at least $2k-1$ and at most $6k-2$ nodes as follows. Consider an arbitrary spanning tree $\mathcal{T}$ of $\mathcal{D}$, and recursively partition $\mathcal{T}$ into subtrees by deleting an edge incident to a centroid node, until any further partition would produce a subtree with less of than $2k-1$ nodes (see Figure 3).

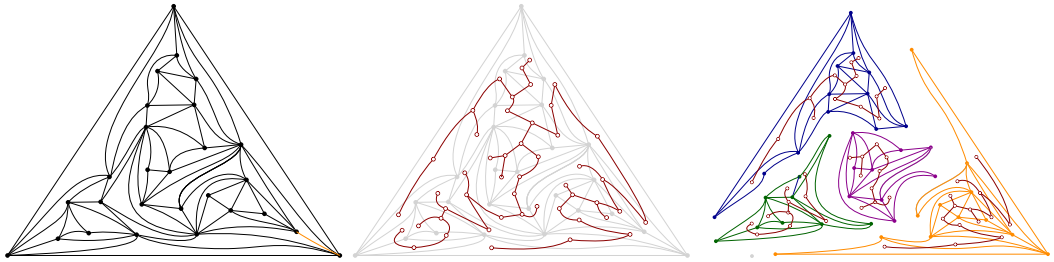


Figure 3 A triangulation on $n = 27$ vertices (a); a spanning tree $\mathcal{T}$ of the dual graph $\mathcal{D}$ (b); four clusters obtained for $k = 5$ (c).

For every $i = 1 \ldots, t$, let R_i denote the region formed by the union of triangles in C_i; and let $V_i \subset V$ denote the set of all vertices of the triangles in C_i. By Euler's formula, applied for the triangles in C_i, we have $k \leq |V(C_i)| \leq 6k$. We call the vertex sets $V_1, \ldots, V_t$ *clusters*, We also define a *cluster graph* $\mathcal{G}$, where the nodes correspond to the t clusters, and nodes V_1 and V_j are adjacent in $\mathcal{G}$ if C_i and C_j contain two triangles that share an edge. Note that a vertex $v \in V$ may be part of arbitrarily many clusters (e.g., if v is a high-degree vertex), yet each cluster is adjacent to $O(k)$ other clusters in $\mathcal{G}$.

We can augment G as follows. (1) Augment each cluster V_i to a clique, drawing all new edges along shortest paths in the region R_i. (2) For each pair of adjacent clusters, V_i and V_j, add a matching of size $k - |V_i \cap V_j|$ between $V_i \setminus V_j$ and $V_j \setminus V_i$, drawing the new edges along shortest paths in the region $R_i \cup R_j$. Let G' denote the augmented graph.

First, we show that G' is k-connected. Specifically, we show by induction on $i \geq 1$ that if i clusters induce a connected subgraph of the cluster graph, then the union of the clusters induce a k-connected graph in G'. The claim is clear in the base case $i = 1$. For the induction step, we use Menger's Theorem. Assume that the claim holds for $i-1$ clusters. The union of i clusters is composed of two parts, A and B, where A is the union of $i-1$ clusters, including cluster V_j, and $B = V_i$ is the i-th clusters. Both $G'[A]$ and $G'[B]$ are k-connected by the induction hypothesis, they share $|V_i \cap V_j| \geq 2$ vertices, and are connected by a matching of size $k - |V_i \cap V_j|$ (which is disjoint from the shared vertices). Let $a_1 = b_1, \ldots, a_s = b_s$ be the shared vertices, and let $a_t b_t$, $t = s+1, \ldots k$, be the matching with $a_i \in A \setminus B$ and $b_i \in B \setminus A$. Let $a \in A$ and $b \in B$ be arbitrary vertices. By Menger's Theorem, there are k internally disjoint paths from a to $a_1, \ldots, a_k$ in $G'[A]$, hance k internally disjoint path from a to $b_1, \ldots, b_k$ in $G'[A \cup B]$. Since $G'[B]$ is a clique, we obtain k internally disjoint paths between a and b in $G'[A \cup B]$. By Menger's Theorem, $G'[A \cup B]$ is k-connected.

Second, we claim that G' is $O(k^2)$-planar. Consider an edge e in a cluster C. In the complete graph of C, edge e is crossed by $O(k^2)$ other edges. Cluster C is adjacent to $O(k)$ other clusters, each of which is connected to C by a matching of size $O(k)$. Edge e crosses

$O(k^2)$ edges in these matchings. Therefore, any edge in a cluster crosses $O(k^2)$ other edges in G'. Now consider an edge f of a matching between two adjacent clusters A and B. Then f can cross any of the $2 \cdot O(k^2)$ edges in the two cliques induced by clusters A and B. It can also cross the $2 \cdot O(k^2)$ edges in the $O(k)$ matchings between A (resp., B) and their adjacent clusters. Overall, any edge in G' has $O(k^2)$ crossings. ◀

4 $1 \to 3$ Connectivity Augmentation for Plane Trees

In this section we present a linear-time algorithm for the following problem: Given an n-vertex plane tree T (i.e., with a fixed embedding), compute a 3-connected plane graph G with the minimum number of edges such that T is a spanning subgraph of G. That is, we solve the 3-connectivity augmentation of plane trees in the topological setting in linear time. We remark that Dhanalakshmi, Sadagopan, and Manogna [18] solve the problem for a tree T, where G is not necessarily planar. However, we could not fully verify the correctness of their algorithm. In their last step, the algorithm seems to introduce a matching on a long path of degree-2 vertices in T, which does not result in a 3-connected graph.

Let us start with a simple lower bound (which also appears in [18]) on the number of inserted edges. For a graph $G = (V, E)$ and integer $i \geq 0$, let $n_i(G) = |\{v \in V \colon \deg(v) = i\}|$ denote the number of vertices in G of degree i.

▶ **Observation 4.1.** *Let T be a spanning tree in a 3-connected graph G. Then G has minimum degree $\delta(G) \geq 3$. In particular, $|E(G)| - |E(T)| \geq \left\lceil \frac{2n_1(T)+n_2(T)}{2} \right\rceil$.*

We present an algorithm that meets the lower bound of Observation 4.1.

▶ **Theorem 4.2.** *Let T be an n-vertex plane tree with $n \geq 4$ and $n_i(T)$ vertices of degree i. Then there exists a plane n-vertex graph G with $T \subseteq G$ and $|E(G)| - |E(T)| = \left\lceil \frac{2n_1(T)+n_2(T)}{2} \right\rceil$.*

Moreover, such a G can be computed in $O(n)$ time.

Proof. We present our construction of G as an inductive proof on n, which can be easily turned into a linear-time recursive algorithm. For simplicity, we call the edges of $E(G) \setminus E(T)$ the *new* edges and illustrate them in red. For the induction base of $n = 4$, we set $G = K_4$. So let us henceforth assume that $n \geq 5$. We use a stronger induction hypothesis, where we require that $n_2(T)$ is even; we later show how to handle the odd case. This assumption ensures that each leaf of T is incident to exactly two new edges, each vertex of degree 2 to exactly one new edge, and all other vertices to no new edges. We distinguish four cases.

Case 1: A vertex v has two consecutive neighbors a, b of $\deg(a) = \deg(b) = 1$.

Say a comes clockwise after b at v. We remove a and b from T. If v is not a leaf after this removal, then we add another leaf w at v; otherwise v takes the role of w. We denote the new tree by T' and note that $n_2(T') = n_2(T)$, i.e., the number of degree-2 vertices remains the same. In order to apply the induction, we must also ensure that T' has at least 4 vertices if no new leaf was added, i.e., T has at least 6 vertices. This is the case as otherwise T corresponds the tree T_5^* depicted in Figure 5, for which $n_2(T_5^*)$ is odd. In the obtained solution G' with $T' \subseteq G'$, vertex w is incident to two edges in $F' = E(G') - E(T')$, say wx comes clockwise after wy at w. Then we remove wy and insert ab and by.

We define a *leg* as a vertex v in T of $\deg(v) = 2$ with neighbor a of $\deg(a) = 1$.

Case 2: A vertex v has two neighbors a, b that are legs which are either consecutive or have a leaf c between them.

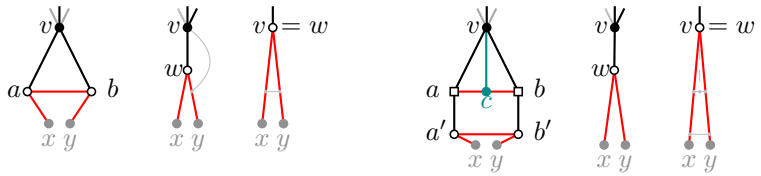


Figure 4 Illustration of Case 1 and 2 in the proof of Theorem 4.2.

We may assume that a comes clockwise after b, and if it exists, c comes after b and before a at v. Then we remove a, b, their adjacent leaves a', b' and c. If v is not a leaf after this removal, then we add another leaf w at v; this ensures that v does not have degree 2. As before, we aim to recurse on the resulting tree T'. Note that $n_2(T') = n_2(T) - 2$ because a and b are removed and v has degree 1 or at least 3 in T'. We also ensure that T' has at least 4 vertices. In the case that we introduce a new leaf w, this is due to the fact that v has at least two more neighbors. In the other case, we must exclude the case that T' is a path on three vertices. However, this would imply that T is either the tree T_7^* or T_8^*, illustrated in Figure 5, both of which have an odd number of degree-2 vertices. Thus, we may apply induction on T'.

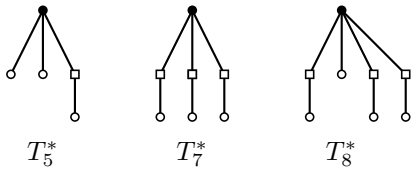


Figure 5 Illustration for the (non-existing) base cases in the proof of Theorem 4.2.

In the obtained solution G', vertex w has two incident edges in $F' = E(G') - E(T')$, say wx comes clockwise after wy at v. Then we remove wx, wy and insert $a'x$, $a'b'$, and $b'y$. If c exists in T, we insert ac and cb, otherwise we insert only ab.

Case 3: A vertex v with $\deg(v) = 2$ has a neighbor b that is a leg adjacent to leaf a.

In this case we delete a and b and recurse on $T' = T - \{a, b\}$; note that $n_2(T') = n_2(T) - 2$. Moreover, if T' had only 3 vertices, then T would correspond to a path on 5 vertices for which $n_2(T)$ would be odd. Hence, we may apply the induction. In the obtained solution G', vertex v has two incident edges in $F' = E(G') - E(T')$, say vx and vy. Then we remove vx, vy and insert va, ax, and by.

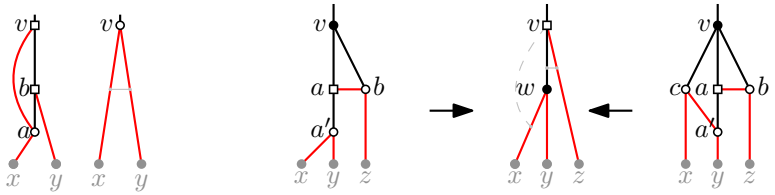


Figure 6 Illustration of Case 3 and 4 in the proof of Theorem 4.2.

Case 4: A vertex v (with $\deg(v) = 3$ or 4) has a neighbor p with $\deg(p) \geq 3$, one neighbor a which is a leg and otherwise only leaves.

By Case 1, we may assume that no two leaves are consecutive. Hence, either v has exactly one leaf b or two leaves b, c which are separated by p and a, see also Figure 6. In this case we delete a, b, c and the leaf a' of a, and insert a new leaf w at v. We recurse on the resulting tree T'. Note that $n_2(T') = n_2(T)$ as we deleted a and v has degree 2 in T'. Moreover, due to the fact that p has $\deg(p) = 3$, T' has at least 4 vertices. In the solution G' obtained by induction, vertex v is incident to one edge, say vz, and vertex w to two edges, say wx and wy, in $F' = E(G') - E(T')$. Then we remove wx, wy, wz and insert $a'y$, ab, and bz. If c does not exist, we insert $a'x$ and otherwise $a'c, cx$.

This completes the construction of G with $T \subseteq G$. To show that the case distinction is exhaustive, root the tree T at an arbitrary vertex and consider a lowest vertex v that is neither a leaf nor a leg. All but one neighbor p of v, are either leaves or legs. If v has $\deg(v) = 2$, then Case 3 applies. Otherwise, apart from p, v has two consecutive leaves (Case 1), two legs which are consecutive or have a leaf between them (Case 2), or one leg and otherwise only leaves (Case 4).

It remains to argue that the resulting graph G is 3-connected. This is clear in the base case $G = K_4$. In each remaining case, it suffices to observe that G is obtained from a 3-connected graph G' by a local modification that is a sequence of so-called *BG-operations*; these operations were introduced by Barnette and Grünbaum [8] and preserve 3-connectivity. In a $(1, 2)$-*operation* we subdivide an edge xy by a new vertex w and insert an edge vw with $v \neq x, y$. In a $(2, 3)$-*operation* we subdivide edge xy by vertex a and edge vw by vertex b and insert the edge ab. Figures 4 and 6 illustrate how up to three of these operations suffice.

Lastly, we show how to handle the case that $n_2(T)$ is odd. We choose an arbitrary vertex v of degree 2 and contract one of its incident edges to obtain a tree T'. Using our above machinery, we obtain a 3-connected graph G' with $|E(G')| - |E(T')| = \left\lceil \frac{2n_1(T)+n_2(T)-1}{2} \right\rceil = \left\lceil \frac{2n_1(T)+n_2(T)}{2} \right\rceil - 1$. Now we reintroduce v by a $(1, 2)$-operation on its edge, connecting v to an arbitrary non-neighbor in an incident face. As this preserves 3-connectivity, this completes the proof. ◀

5 Hardness Results

In this section we prove our main hardness results (Theorem 1.1 and Corollary 1.3). We reduce from planar 3-SAT. A 3-SAT instance is given by a boolean 3-CNF formula φ, with n variables $x_1, x_2, \ldots, x_n$ and m clauses $C_1, C_2, \ldots, C_m$. In planar 3-SAT, φ can be represented as a planar bipartite graph with vertices for every variable and clause, and an edge (x_i, C_j) if x_i or $\neg x_i$ is a literal in clause C_j. This is a well studied SAT variant proven NP-complete by Lichtenstein [41]. We have three reductions that respectively cover the $2 \to 3$, $3 \to 4$ and $4 \to 5$ augmentations. Note that if the $c \to k$ augmentation problem is NP-complete, so is the $c' \to k$ problem for $c' < c$ because every c-connected graph is also c'-connected. All three reductions are similar: constructed from three gadgets (variable, clause, and literal) that are put together in the standard way. Due to the similarities between reductions, we will address only one of them here and relegate the other two to the full version of this paper. In particular, we only provide our reduction for $3 \to 4$ augmentation as it is the best representative of the three reductions.

5.1 $3 \to 4$ Connectivity Augmentation is NP-Complete

Our reduction works for all settings (abstract, topological and geometric) since 3-connected graphs have a unique combinatorial embedding and the reduction can be embedded geometrically. We use three gadgets, a variable, a literal, and a clause gadget. We ensure that

the entire construction is 4-connected except for some select degree-3 vertices. The variable gadget is shown in Figure 7b. The "building block" of our construction is essentially a wheel graph on four vertices W_4. We modify this W_4, removing two edges so that one vertex is degree-3, one is degree-1, and two are degree-2 (see Figure 7a). We call the degree three vertex the *central* vertex and the others the *boundary* vertices. We construct a variable gadget by "gluing" an even number of these modified W_4's together so any two consecutive W_4 have two shared boundary vertices. Hence every boundary vertex is identified with two separate W_4's, while the central vertices are unique to each W_4, so every central vertex is precisely degree three and must therefore be augmented.

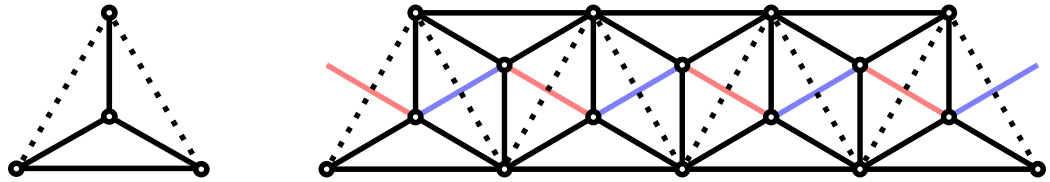


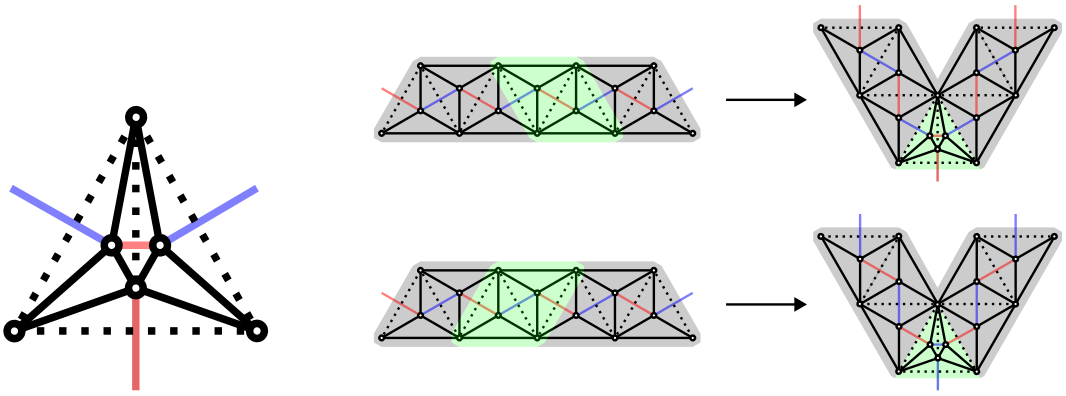
Figure 7 (a) Modified W_4, removed edges indicated with dashed line. (b) A variable gadget. Red segments indicate a false assignment, blue indicate a true assignment.

▶ **Lemma 5.1.** *The variable gadget (Figure 7b) constructed with $2k$ modified W_4's can be minimally augmented with k edges, and there are two such augmentations.*

Proof. As each central vertex is still unique to its W_4, the boundary vertices of each W_4 form a 3-cut disconnecting their central vertex. Therefore, augmenting this gadget to be 4-connected requires adding an edge from each central vertex to a vertex in a separate W_4. This augmentation must be planar and so each central vertex can only be connected to the central vertices of its two adjacent W_4's. Note, once a central vertex is connected to another central vertex, each of their adjacent boundary vertices no longer form a 3-cut. So we only need to connect every other pair of central vertices, and in fact this would be the minimal augmentation for this gadget. Each central vertex must be connected to at least one other vertex, and they can only be connected to other central vertices. Therefore connecting each central vertex to precisely one central vertex must be minimal, and clearly the number of edges required is precisely half the number of central vertices.

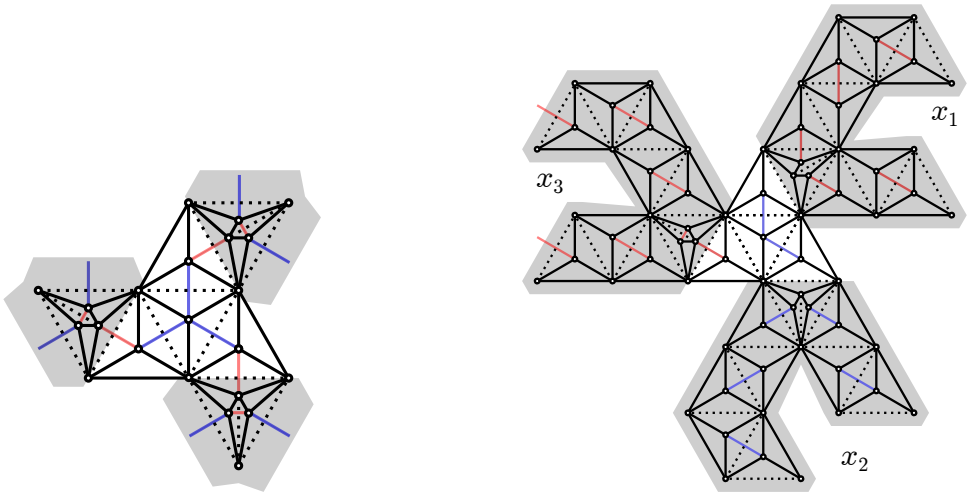
Note there are two unique ways to do this. Arbitrarily identify one W_4 as the "initial" and orient the variable cycle from this W_4. The two possible augmentations are therefore either each central vertex is attached to its clockwise neighbor, or each central vertex is adjacent to its counter clockwise neighbor. If clockwise neighbors are adjacent we say the variable is set to "true" and we say it is "false" if counter clockwise neighbors are connected. ◀

A literal gadget (Figure 8a) is composed of six vertices, and similar to the modified W_4's used in the variable gadget. Again, as with the variable gadget, the three boundary vertices form a 3-cut that disconnects the three central vertices. As depicted in Figure 8b literal gadgets are "inserted" into a variable gadget, essentially replacing two of W_4's of a variable. Consider a literal gadget inserted into a variable x_i. There are only two possible locally minimal edge augmentations of the literal gadget, as depicted in Figure 8b. For a positive literal (Figure 8b(top)) if x_i is true (resp., false) we connect the degree-3 vertices with the blue (resp., red) matching.



(a) (b)

Figure 8 (a) A positive literal gadget (b) How a literal gadget is "inserted" into a variable gadget. Two W_4's of the variable are replaced by one literal. A clause gadget is adjacent to the exposed dotted line. On the top is a positive literal, the bottom shows a negated literal.



(a) (b)

Figure 9 (a) A clause gadget (b) Example where two variables are set to true and one is set to false. Variables are highlighted in grey.

The clause gadget (Figure 9) requires 3 new edges, and is achievable iff a literal is true.

▶ **Lemma 5.2.** *The clause gadget shown in Figure 9 can be augmented with exactly three edges if and only if one adjacent literal gadget is true. Otherwise 4 edges are necessary.*

Proof. Let c be a clause of φ on three variables x_i, x_j, and x_k. For each literal of c we "glue" a term K_4 to the corresponding variable gadget. We take two adjacent W_4's of the variable gadget and replace them with a literal gadget. (Figure 8b).

To augment the clause gadget, for each of its W_4's the central vertex must be connected to a vertex of another W_4. Note, the central vertex of the value W_4 can only connect to one of the central vertices of the terms. While each term W_4 can either have its central vertex connected to a vertex in an adjacent literal gadget or to the central vertex of the value W_4.

Therefore, augmenting this gadget requires at least three edges, one edge for each term. We now show that this can be augmented exactly three edges only if at least one adjacent literal gadget is true.

If a term is adjacent to a false literal then the only possible minimal augmentation is to connect the central vertex of the term to a central vertex of the literal gadget. This would augment both the term and the literal by adding only one edge. Any other augmentation would require at least two edges, we would have to connect a central vertex of the literal to a variable W_4 and we would have to connect the central vertex of the term to the central vertex of the value of the clause.

Note that a true literal gadget is already 4-connected. Therefore, if a term W_4 is adjacent to a true literal then we can connect its central vertex to the central vertex of the value of the clause, augmenting both to be 4-connected.

Therefore we can augment a clause gadget with only 3 edges if and only if it is adjacent to a true literal gadget. Otherwise we require 4 edges. If every term is adjacent to a false literal, then each of the false literal gadgets require one edge and the value of the clause requires an edge. As the augmentation must be planar, these edges are disjoint, and so we need to add four edges. So, c can be augmented with three edges if and only one adjacent literal gadget is true. ◀

With these three gadgets we can reduce from planar 3-SAT, giving the following lemma.

▶ **Lemma 5.3.** *Given a 3-connected planar graph $G = (V, E)$ and an integer τ, it is NP-hard to decide whether there is a set F of edges with $|F| \leq \tau$ such that $G' = (V, E \cup F)$ is a 4-connected planar graph.*

Proof. Recall that we consider the decision version of this problem. I.e. given a 3-connected plane graph G, can at most τ edges be added to form a 4-connected plane graph G'?

Given a planar 3-SAT formula φ with n variables and m clauses. Let P be a planar embedding of φ. We now construct an instance of 3-4 graph augmentation, starting with a 3-connected plane graph.

Take every variable's embedding in P each will form a star graph. For each variable we create a corresponding variable gadget by taking a Eulerian circuit of each of these stars and replacing it with a variable gadget. These gadgets are cycles of an even number of modified W_4's. Next replace every clause node with a clause gadget and for each of the three literals insert corresponding literal gadgets into the relevant variable gadgets, replacing two W_4's. We can then compute what the correct value of k is using Lemmas 5.1 and 5.2. That is, k is half the number of W_4's that occur in all variable gadgets plus 3 for each clause gadget $(3m)$.

Now given a satisfying assignment S for φ we can augment each variable gadget accordingly and then if S is a satisfying assignment each clause gadget should have an adjacent true literal gadget. Therefore by Lemma 5.2 each clause can be augmented with 3 edges. So G can be augmented with at most τ edges.

If there is a set of τ edges, where τ is half the number of variable W_4's plus 3 times the number of clauses that planarly augments G. Then there is a corresponding satisfying assignment of φ. By Lemma 5.1 there are only two ways to minimally augment each variable gadget, corresponding to false or true assignments respectively. Therefore, each variable gadget will uniquely encode a corresponding variable assignment, this gives some possible solution S' for φ. By Lemma 5.2 we will only be able to augment each clause gadget with three edges if one of its adjacent literal gadgets is true, otherwise at least one will require 4 edges. So we will only have an augmenting set of τ edges if every clause of φ has at least one true literal. So if there is an augmenting set of τ edges we can get a satisfying assignment S' for φ by checking each variable gadget.

We conclude that φ has a satisfying assignment S if and only if G can be augmented with at most τ edges. ◀

▶ **Corollary 5.4.** *Lemma 5.3 still holds if G is a topological (or geometric) plane graph and if G' is required to be a compatible topological (resp., geometric) plane or 1-plane graph.*

Note, every argument that was made was independent of the magnitude or drawing of any edge (as long as they are planar). Hence our reduction easily extends to the topological and geometric settings, by just embedding each gadget (and therefore G) as depicted. The reduction is also easily adapted to the 1-planar setting. In each of our gadgets, we included dashed lines, like the ones between every W_4 in the variable gadget. By replacing these with an edge in G, augmenting G will necessarily result in a 1-planar graph.

▶ **Corollary 5.5.** *Given a combinatorial triangulation T and an integer τ. Determining if there is a series of $\leq \tau$ flips that transforms T into a 4-connected triangulation T' is NP-complete.*

This is a simple extension of our main reduction. Note, every black dotted line corresponds in our gadgets (Figure 7, Figure 8, and Figure 9) were an edge of G. Then G is a combinatorial triangulation, and each augmenting edge can be realized by a flip of one of these dotted edges. After adding these dotted lines as edges of G, our arguments from the $3 \to 4$ augmentation reduction easily translate.

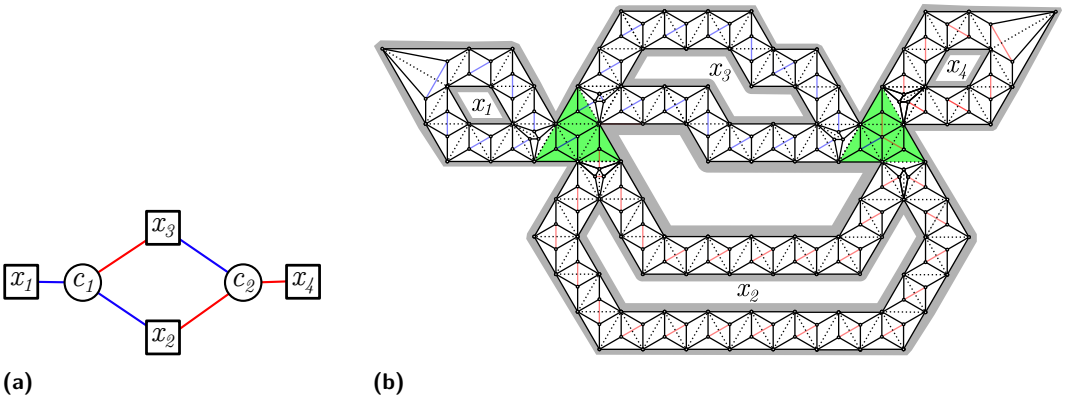


Figure 10 (a) A planar 3-SATformula $\varphi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)$ (b) The graph G constructed from φ during the reduction, x_1 and x_3 are assigned true, and x_2 and x_4 are set to false. Clause gadgets are highlighted in green for legibility.

6 EPTAS for Flip Distance of Triangulations to 4-Connectivity

In this section, we provide an EPTAS for making plane triangulations 4-connected with a sequence of edge flips.

▶ **Theorem 6.1.** *For every $\epsilon > 0$, there is an $n \cdot 2^{O(1/\epsilon)}$-time algorithm that, for a given triangulation G on $n \geq 6$ vertices, returns a sequence of edge flips that turns G into a 4-connected triangulation, and the length of the sequence is at most $1 + \epsilon$ times larger than necessary.*

We first reduce the problem to a certain hitting set problem as suggested by Bose et al. [13]. The reduction relies on the following.

▶ **Lemma 6.2** (Mori et al. [42]). *Let G be a plane triangulation on $n \geq 6$ vertices, T a separating triangle of G and e an edge of T. Then flipping e destroys the separating triangle T and does not create any new separating triangles in G, provided e is incident to multiple separating triangles or none of the three edges of T are incident to multiple separating triangles.*

This allows one to easily reduce the problem of finding an optimal sequence of flips to a certain hitting set problem on the edges of the triangulation.

▶ **Lemma 6.3.** *Let G be a plane triangulation on $n \geq 6$ vertices. Let E' be a set of edges of G such that every separating triangle in G is incident to at least one edge in E', and let τ be the minimum size of any such set.*

Making G 4-connected requires at least τ edge flips. Moreover, given E', one can compute a sequence of edge flips of length at most $|E'|$ making G 4-connected in $O(n)$ time.

Proof. If F is a sequence of edge flips making G 4-connected, then every separating triangle in G must be incident to at least one edge in E' (otherwise the missing separating triangle remains in G after all flips have been performed, and the graph is not 4-connected). Thus, F has length at least τ.

Now consider the edges of E' in some arbitrary order $e_1, e_2, \ldots, e_{|E'|}$. Define a sequence of edge flips as follows. Let $1 \leq i \leq |E'|$, and assume the first $i-1$ edges of E' have already been processed. Then for the e_i, do the following:

- if e_i is not present in G (because it has already been flipped previously) or does not bound any separating triangles in G, don't flip any edge;
- otherwise, if e_i is incident to multiple separating triangles in G, flip e_i;
- otherwise, if none of the 3 edges of the unique separating triangle T incident to e are incident to any other separating triangle, flip e_i;
- otherwise, flip an edge of T incident to at least 2 separating triangles.

By the definition of this sequence and of E, it is clear that for any separating triangle in G, at least one of its 3 edges gets flipped. Moreover, every edge that is flipped obeys the conditions of Lemma 6.2 (at the time it is flipped), and thus, no new separating triangle is introduced and the resulting graph is 4-connected.

Let us describe how to carry out the procedure in linear time. For each edge, we will maintain a set of pointers to the separating triangles it bounds in a doubly-linked list. Each separating triangle also has corresponding back-pointers, so that from a separating triangle we can access the three pointers pointing to it in constant time. We can initialize this structure by computing the set of separating triangles incident to each edge in $O(n)$ total time (this can be done for example by computing a 4-block decomposition of the triangulation in linear time [36]).

For a given edge, we can then test in constant time if it bounds 0, 1 or at least two separating triangles. In the case it bounds exactly one separating triangle T, we can test in constant time whether this is also the case for the two other edges bounding T. Each time we flip an edge e, we go through the set of separating triangles it used to bound by traversing its associated doubly-linked list and following the pointers it stores. For each such separating triangle, we delete it from the doubly-linked lists of the three edges which (used to) bound it by following the back-pointers. This costs constant time per deleted separating triangle.

Because no new separating triangle is created at any point, every separating triangle is deleted once and the triangulation starts out with $O(n)$ separating triangles, the total runtime is $O(n)$. ◀

It is tempting to try to carry out the reduction by simply flipping the edges of E' in some arbitrary order (or perhaps skipping an edge if it is no longer incident to any separating triangle by the time it is considered). However, this strategy may fail as illustrated in Figure 11. Thus, a more careful strategy is needed. Our algorithm flips only edges that bound separating triangles and do not create any separating triangle, thus strictly decreasing the number of separating triangles with each flip.

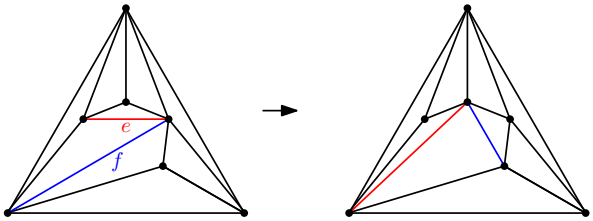


Figure 11 While $\{e, f\}$ is a minimum set of edges hitting every separating triangle, flipping e then f fails to make the triangulation 4-connected, as a new separating triangle is introduced.

▶ **Lemma 6.4.** *Given a plane graph G of treewidth $\leq k$ with n vertices, and a set C of 3-cycles of G, there is a $n \cdot 2^{O(k)}$-time algorithm that computes a minimum cardinality edge set E' such that every 3-cycle in C is incident to at least one edge in E'.*

Proof. Recall that a tree decomposition of G is a mapping of G into a tree T_G whose vertices we shall call *bags*. A bag B is associated with a subset of vertices V_B. Every vertex appears in at least one bag and induces a single subtree in T_G, and if two vertices are adjacent in G then there is a bag in T_G containing both vertices. The width of T_G is the size of its largest bag. We root T_G arbitrarily. We can build a tree decomposition of G of width $O(k)$ in $n \cdot 2^{O(k)}$ time with the following properties [11, 39]:

- the number of bags is $O(n)$;
- the maximum degree of a bag is 3;
- for a bag with two children, all three bags have identical subsets of vertices; and
- for a bag with a single child the two subsets differ by one.

Since G is planar, each bag defines an induced subgraph with $O(k)$ edges. Note that, by definition, if G has a clique, these vertices must appear in a bag of T_G. Thus, every 3-cycle in C must appear in at least one bag of T_G. We compute E' using dynamic programming as follows. We say that a subset S of edges *satisfies* a subtree of T_G if every 3-cycle of C that appears in a bag of the subtree contains at least one edge in S. For each bag B and for each subset S of the $O(k)$ edges in V_B, we define a subproblem $A_{B,S}$ of computing the minimum cardinality set S' of edges satisfying the subtree rooted at B subject to choosing the edges in S in the induced subgraph of V_B. Thus there are $n \cdot 2^{O(k)}$ total subproblems. By our degree-3 assumption, the children of parents of high-degree simply receive the information from their parent without making any choices. The child of a low degree bag has to make choices about the incident edges to the potential new vertex v. The degree of v in the induced subgraph is $O(k)$ which implies that we have to make $2^{O(k)}$ binary choices. Some of these choices are fixed: if the addition of v closes a 3-cycle in C, the cycle has two incident edges to v and the choices are valid only if at least one edge of the separating triangle is chosen. Therefore to compute each subproblem there is $k \cdot 2^{O(k)} = 2^{O(k)}$ nonrecursive work. Since the tree decomposition has $O(n)$ nodes, this approach leads to a $n \cdot 2^{O(k)}$ runtime. ◀

We can now give the proof of Theorem 6.1.

Proof of Theorem 6.1. The idea is to apply Baker's layer shifting technique [7] to the hitting set formulation of the problem. We actually solve the more general problem of hitting an arbitrary subset C of 3-cycles of G, given as an input.

Assume without loss of generality that $\epsilon \leq 1/2$ and set $k = \lceil 1/\varepsilon \rceil$. Choose an arbitrary vertex v_0 of G and label all vertices of G according to their (shortest path) distance to v_0. We call the set of vertices at a specific distance from v_0 a layer of G. For $0 \leq i < k$, assign the color i to all edges incident to two vertices labeled (i mod k) (edges joining two vertices in consecutive layers do not get assigned a color).

Let E^* be a smallest subset of edges such that every 3-cycle in C is incident to at least one edge in E^*. Notice that there is at least one of the k colors, say, c, for which at most $|E^*|/k$ edges from E^* get assigned the color c.

Now imagine decomposing G into the subgraphs $G_0, \dots G_p$ each induced by $k+1$ consecutive layers, starting at a layer whose label is (c mod k) (except eventually for the first subgraph which starts at layer 0, and the last subgraph which might consist of fewer than $k+1$ consecutive levels). Note that consecutive subgraphs overlap at layers (c mod k). Every 3-cycle in C appears in one of the subgraphs, as it necessarily has its 3 vertices in at most 2 consecutive layers, and every pair of consecutive layers appears in one of the subgraphs. Taking the union of optimal solutions for the problem on each subgraph (with C restricted to the 3-cycles which appear in that subgraph) thus gives a valid solution for G. Let $E_1, \dots, E_p$ be such optimal solutions, and for $1 \leq j \leq p$ let $E^*[G_j]$ denote the subset of edges in E^* which appear in G_j. We have that

$$\left| \bigcup_{1 \leq j \leq p} E_j \right| \leq \sum_{1 \leq j \leq p} |E_j| \leq \sum_{1 \leq j \leq p} |E^*[G_j]|$$
$$\leq |E^*| + |E^*|/k \leq (1+\epsilon)|E^*|,$$

where the second inequality stems from the fact that $E^*[G_j]$ is a valid solution for G_j, the third from the fact that only the edges of color c get counted twice (and there are at most $|E^*|/k$ such edges) and the last from the choice of $k = \lceil 1/\epsilon \rceil$.

The graphs $G_0, \dots, G_p$ are all $(k+1)$-outerplanar and thus have treewidth $O(k)$ [10]. Using Lemma 6.4, we can thus solve the problem optimally for each G_j in $|V(G_j)|2^{O(1/\epsilon)}$ time. The total runtime is then $\sum_{1 \leq j \leq p} |V(G_j)|2^{O(1/\epsilon)} = n \cdot 2^{O(1/\epsilon)}$. We can carry out the described procedure for each value of $1 \leq c \leq k$ (note that we don't know the right value of c to choose *a priori*) and take the best solution. This adds a factor of $1/\epsilon$ that gets absorbed by the exponential. The previous discussion, together with Lemma 6.3, then gives the sought result. ◀

While Baker's method of decomposing a planar graph into overlapping layers of small treewidth and combining the solutions is classical, it is interesting to note that a direct adaptation fails for the problem of approximating a smallest set of edges which make a triangulation 4-connected when flipped simultaneously (or the equivalent problem of augmenting a triangulation to 1-planar 4-connected via edge insertions). This is because, while this problem is still efficiently solvable for small treewidth, taking the union of solutions might lead to selecting edges which are not simultaneously flippable. Perhaps this could be overcome by combining the solutions more carefully, but we leave the problem of finding a PTAS for the simultaneous case open here.

Previous results. Accornero, Ancona and Varini [2] present an algorithm and claim it computes the minimum number of edges hitting all separating triangles in a given triangulation. Note that this result together with our reduction in Section 5 (Corollary 1.3) would imply

P=NP. Although their algorithm computes a set of edges that hits all separating cycles, we claim this set is not minimum. We attempt here to give a short intuition for that. In short, their algorithm computes (the equivalent of) a 4-block tree of the input triangulation T, a decomposition of T into maximal 4-connected components (4-blocks) hierarchically organized as the nesting structure of separating triangles that bound the maximal components. At each level, the edges are weighted with the cost of the optimal solution for the children nodes that contain said edge in their outer face. Note that every edge bounds at most two separating triangles in the next level of the tree and, therefore, there are at most two children associated with the same edge. At the root of the tree (and recursively in every node) they then compute the minimum weight subset of edges that hit every separating triangle in the maximal 4-block associated with the node. In [2, Theorem 3] there is an implicit assumption that the optimal solution contains a single edge of each separating triangle, since the weights are computed based on the cost of recursive solutions that flip a specific edge. That is not true, for example, in our reduction where the optimal solution may flip two edges of a separating triangle bounding the literal gadget (blue edges in Figure 8).

7 $2 \to 3$ Augmentation for Geometric Triangulations of n Points in Convex Position

Given a triangulation $G = (V, E)$ of n points in convex position and a positive integer ℓ, we can compute the *minimum* number of edges that augment G to a 3- or 4-connected ℓ-planar graph (or report that there is no such augmentation) by dynamic programming.

The key observation is a simple combinatorial characterization of 3- or 4-connected augmentation. We introduce some terminology. An edge $e \in E$ is a *diagonal* if it is not an edge of the convex hull of V; and $e \in E$ is an *ear* if there is exactly one point in V in an open halfplane bounded by the line spanned by e.

▶ **Proposition 7.1.** *Let $G = (V, E)$ be a straight-line triangulation on n points in convex position, and let $G' = (V, E \cup F)$ be a geometric graph. Then*

- *G' is 3-connected if and only if every diagonal in E crosses an edge in F;*
- *G' is 4-connected if and only if every diagonal in E crosses at least two edges in F; and every diagonal in E is either an ear or crosses at least two disjoint edges in F.*

Proof. If G' is 3-connected, then the two endpoints of a diagonal $e \in E$ cannot be a 2-cut, and so some edge $f \in F$ connects the two components of $G - ab$. Since the vertices are in convex position, then e and f cross.

Conversely, suppose that every diagonal in E crosses an edge in F, and G' has a 2-cut $\{a, b\}$. Then $\{a, b\}$ is already a 2-cut in G, so ab is a diagonal, and $G - ab$ has exactly two connected components. However, an edge $f \in F$ that crosses ab connects the two components, contradicting the assumption that $\{a, b\}$ is a 2-cut in G'.

If G' is 4-connected, then the two endpoints of a diagonal $ab \in E$ and an endpoint of a crossing edge $cd \in F$ cannot form a 3-cut. This leaves us with two possibilities: either c or d is the only vertex in one of the components of $G - ab$ (in this case ab is an ear), or there an edge in $f_1 \in F$ that crosses ab and not incident to c, and an edge $f_2 \in F$ that crosses ab and not incident to d. Now if $f_1 = f_2$, then cd and $f_1 = f_2$ are disjoint, otherwise f_1 and f_2 are disjoint. In both cases, two disjoint edges in F cross ab.

Conversely, suppose that G' satisfies the conditions above but has a 3-cut $\{a, b, c\}$. If none of ab, ac and bc is diagonal in E, then $G - \{a, b, c\}$ is connected, so $\{a, b, c\}$ would not be a 3-cut. First, assume that ab, ac, and bc are all diagonal in G. Then $G - \{a, b, c\}$ has exactly

three components. However each edge of $\Delta(a,b,c)$ crosses two edges in F, and in particular at least one edge that is not incident to $\{a,b,c\}$. Thus, each of the three components of $G-\{a,b,c\}$ is connected to another component in G', and so $G'-\{a,b,c\}$ is connected.

Next, assume that only one edge of $\Delta(a,b,c)$, say ab, is a diagonal of E. Then $G-\{a,b,c\}$ has exactly two components, separated by the line spanned by ab. In particular, if ab is an ear, then c cannot be the only vertex on one side of the line spanned by ab. In this case, ab crosses at least two edges $f_1, f_2 \in F$. At most one of them is incident to c: This is clear if ab is not an ear, and f_1, f_2 are disjoint. If ab is an ear, then f_1, f_2 are incident to a single vertex on one side of ab, but this vertex is not c, and so one of f_1 and f_2 is not incident to c. Since F contains an edge between the two components of $G-\{a,b,c\}$, then G' is connected. ◀

▶ **Theorem 7.2.** *For any $k \in \{3,4\}$ and any constant $\ell \in \mathbb{N}$, there is an algorithm that, given a PSLG triangulation $G=(V,E)$ on n points in convex position, can find a minimum set F of new edges in $O(n)$ time such that $G'=(V,E\cup F)$ is a k-connected ℓ-planar geometric graph, or reports that there is no such graph G'.*

Proof. Let e_0 be an arbitrary edge of the convex hull of V, and let E_0 be the subset of E comprising e_0 and all diagonals in E. We define a partial order $\preceq$ on E_0: We say that $e \preceq f$ if both f and e_0 are in the same closed halfplane of bounded by the line spanned by e. (In particular, $e \preceq e_0$ holds for all $e \in E_0$. The poset $(E_0, \preceq)$ defines a tree, rooted at e_0, which is binary tree rooted at e_0. For a dynamic programming algorithm, we define the subproblems that correspond to the edge $e \in E_0$.

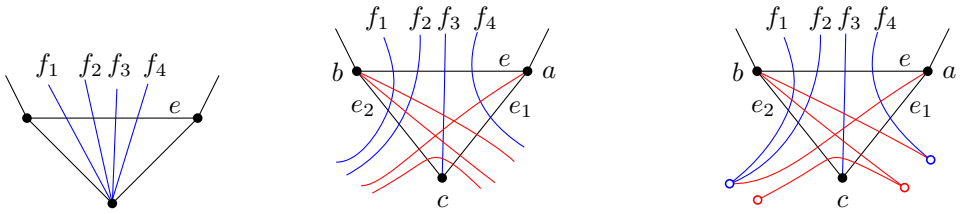


Figure 12 Illustration for a dynamic programming algorithm: An ear diagonal e (a); a non-ear diagonal e for $k=3$ (b), and for $k=4$ (c).

3-connectivity. For an edge $e \in E_0$, integer $q \in \{0,\dots,\ell\}$ and a vector $\vec{c} \in \{1,\dots,\ell\}^q$, let $\mathsf{OPT}_3(e,q,\vec{c})$ denote the minimum size of a set of new edges F such that $G'=(V,E\cup F)$ has the following properties:

- Edge e crosses exactly q edges in F: $f_1,\dots,f_q$;
- for $i=1,\dots,q$, edge $f_i \in F$ crosses at most c_i edges $h \in E_0$, $h \preceq e$; and
- every edge $h \in E_0$, $h \preceq e$, crosses an edge in F (cf. Proposition 7.1 for $k=3$).

If no such set F exists, then we set $\mathsf{OPT}_3(e,q,vecc)=\infty$.

Our DP algorithm considers all edges $e \in E_0$ in increasing order in the poset $(E_0,\preceq)$. If e is a minimal element of the poset $(E_0,\preceq)$, then e is an ear, and $\mathsf{OPT}_3(e,q,\vec{c})=q$ for all $\vec{c}$; see Figure 12a. If e is not minimal, then we may assume that $e=ab$ in a triangle $\Delta(a,b,c)$, where $e_1=ac$ or $e_2=bc$ is also a diagonal in E_0. To compute $\mathsf{OPT}_3(e,q,\vec{c})$, we compare all possibilities in the triangle $\Delta(a,b,c)$ by brute force: Each of the q edges that cross e may cross e_1 or e_2, or end at vertex c. Furthermore, any edge that crosses e_1 (resp., e_2) may end at b (resp., a) or cross e or e_2 (resp., e_1). For each combination, we determine whether they are feasible (i.e., satisfy all constraints), and let $\mathsf{OPT}_3(e,p,\vec{c})$ be the minimum size of a

feasible solution; see Figure 12b. Importantly, the order in which e crosses $f_1, \dots, f_q$ is not specified – we count only unavoidable crossings: The edges in $\{f_1, \dots f_q\}$ that are incident to c must cross all edges that cross both e_1 and e_2, or cross e_1 and end at b, or cross e_2 and end at a. Similar conditions hold for edges that cross e_1 or e_2 by symmetry.

4-connectivity. In this case, the combinatorial characterization in Proposition 7.1 is more involved, and we need to maintain information for bundles of new edges that cross an old edge. For an edge $e \in E_0$, integer $q \in \{0, \dots, \ell\}$, vector $\vec{c} \in \{1, \dots, \ell\}^q$, and a laminar set system P over $\{1, \dots, q\}$, let $\mathsf{OPT}_4(e, q, \vec{c}, P)$ denote the minimum size of a set of new edges F such that $G' = (V, E \cup F)$ has the following properties:

- Edge e crosses exactly q edges in F: $f_1, \dots, f_p$;
- for $i = 1, \dots, p$, edge $f_i \in F$ crosses at most c_i edges $h \in E_0$, $h \preceq e$;
- each set $S \in P$ corresponds to a set $F_{e,S} = \{f_i : i \in S\}$ such that the edges in $F_{e,S}$ have a common endpoint below e or $F_{e,S}$ is the set of all edges in F that cross an edge $h \preceq e$;
- every edge $h \in E_0$, $h \preceq e$, satisfies the conditions in Proposition 7.1 for $k = 4$ assuming the edges in each set $F_{e,S}$, $S \in P$, do not have a common endpoint *above* e.

If no such set F exists, then we set $\mathsf{OPT}_4(e, q, \vec{c}, P) = \infty$.

Our DP algorithm considers all edges $e \in E_0$ in increasing order in the poset $(E_0, \preceq)$. If e is a minimal element of the poset $(E_0, \preceq)$, then e is an ear. In this case, all edges that cross e must have a common endpoint below e: We set $\mathsf{OPT}_3(e, q, \vec{c}, P) = q$ for all $\vec{c}$ if $P = \{\{1, \dots q\}\}$, and $\mathsf{OPT}_3(e, q, \vec{c}, P) = \infty$ for any other partition P; see Figure 12a. If e is not minimal, then $e = ab$ in a triangle $\Delta(a, b, c)$, where $e_1 = ac$ or $e_2 = bc$ is also a diagonal in E_0. We compute $\mathsf{OPT}_4(e, q, \vec{c}, P)$ by a brute force comparison of all possible subproblems for edges e_1 and e_2. By maintaining the laminar system P, we can determine whether every non-ear edge is crossed by at least two disjoint edges. ◀

8 Augmentation for Planar Straight-Line Graphs

As noted in Section 1, the connectivity augmentation problem has been thoroughly studied over PSLGs, where both the input and output graphs are restricted to PSLGs (hence $k \leq 5$). For example, every PSLG G can be augmented to a 2-connected PSLG, and this is the best possible if the vertices are in convex position or an edge of G is a chord of the convex hull. In this section, we relax the planarity constraint for the output graph G'. In general, we would like to determine all pairs of parameters (k, ℓ) such that every PSLG can be augmented to a k-connected ℓ-planar geometric graph.

If we wish to augment a PSLG to a 3-connected geometric graph, we need to go beyond planarity. Our first result addresses the case $k = 3$ for points in convex position.

▶ **Theorem 8.1.** *Every PSLG on n points in convex position can be augmented to a 3-connected 5-planar geometric graph. This bound is the best possible: There exists a triangulation on n points in convex position for which any augmentation to a 3-connected geometric graph yields an edge with at least 5 crossings.*

We can generalize Theorem 8.1 and obtain an asymptotically tight trade-off between connectivity and local crossing number (which matches the bound in Proposition 2.2).

▶ **Theorem 8.2.** *For every $k \in \mathbb{N}$, every PSLG on n points in convex position can be augmented to a k-connected $O(k^2)$-planar geometric graph; and this bound is the best possible.*

Proof. We may assume w.l.o.g. that the input is a triangulation G on n points in convex position. The dual graph is a binary tree with n nodes. We can partition the dual tree into subtrees (*clusters*) of size in the range $[k, 3k]$. Each cluster is a union of triangles, i.e., a convex polygon with $\Theta(k)$ vertices. The adjacency graph of the clusters (*cluster graph*) is a tree of maximum degree $O(k)$.

We can augment G as follows. (1) Augment each cluster to a complete graph; (2) for each pair of adjacent clusters V_1 and V_2 (where $|V_1 \cap V_2| = 2$) add a matching of size $k-2$ between $V_1 \setminus V_2$ and $V_2 \setminus V_1$. Let G' be the augmented graph. Similarly to the proof of Theorem 3.2, the geometric graph graph G' is k-connected (by induction on the clusters), and $O(k^2)$-planar (arguing about edges induced by a cluster, and a pair of adjacent clusters). ◀

For n points in general position, we cannot always augment a PSLG to 4-connectivity with bounded local crossing number.

▶ **Theorem 8.3.** *For every $n \in \mathbb{N}$, there is a straight-line triangulation G on n points such that any augmentation to a 4-connected geometric graph has an edge with $\Omega(n)$ crossings.*

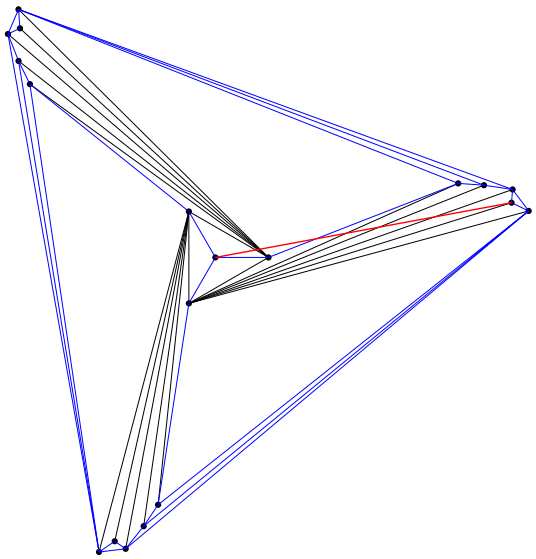


Figure 13 Construction in the proof of Theorem 8.3.

Proof. We construct a straight-line triangulation as follows; see Figure 13. Start with a PSLG K_4, with three points at the vertices of a regular triangle, and one at the center. Attach $(n-4)/3$ long edges to each outer vertex to form a "fan" almost parallel to the three sides of the regular triangle. Let G be an arbitrary triangulation of this graph. In any 4-connected augmentation G' of G, the central vertex of the initial K_4 must be connected to a new vertex. This edge will cross $\Omega(n)$ edges of one of the fans, hence $\Omega(n)$ edges of G. ◀

References

1 Manuel Abellanas, Alfredo García, Ferran Hurtado, Javier Tejel, and Jorge Urrutia. Augmenting the connectivity of geometric graphs. *Comput. Geom.*, 40(3):220–230, 2008. doi:10.1016/J.COMGEO.2007.09.001.

2 Anna Accornero, Massimo Ancona, and Sonia Varini. All separating triangles in a plane graph can be optimally "broken" in polynomial time. *Int. J. Found. Comput. Sci.*, 11(3):405–421, 2000. doi:10.1142/S012905410000020X.

3 Miklós Ajtai, Vašek Chvátal, Monroe M. Newborn, and Endre Szemerédi. Crossing-free subgraphs. In *Theory and Practice of Combinatorics*, volume 60 of *North-Holland Math. Stud.*, pages 9–12. North-Holland, 1982.

4 Hugo A. Akitaya, Rajasekhar Inkulu, Torrie L. Nichols, Diane L. Souvaine, Csaba D. Tóth, and Charles R. Winston. Minimum weight connectivity augmentation for planar straight-line graphs. *Theor. Comput. Sci.*, 789:50–63, 2019. doi:10.1016/J.TCS.2018.05.031.

5 Marwan Al-Jubeh, Mashhood Ishaque, Kristóf Rédei, Diane L. Souvaine, Csaba D. Tóth, and Pavel Valtr. Augmenting the edge connectivity of planar straight line graphs to three. *Algorithmica*, 61(4):971–999, 2011. doi:10.1007/S00453-011-9551-0.

6 Greg Aloupis, Luis Barba, Paz Carmi, Vida Dujmović, Fabrizio Frati, and Pat Morin. Compatible connectivity augmentation of planar disconnected graphs. *Discret. Comput. Geom.*, 54(2):459–480, 2015. doi:10.1007/S00454-015-9716-8.

7 Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994. doi:10.1145/174644.174650.

8 David W. Barnette and Brank Grünbaum. On Steinitz's theorem concerning convex 3-polytopes and on some properties of planar graphs. In G. Chartrand and S. F. Kapoor, editors, *The Many Facets of Graph Theory*, pages 27–40. Springer, 1969. doi:10.1007/BFb0060102.

9 Therese Biedl. A note on 1-planar graphs with minimum degree 7. *Discret. Appl. Math.*, 289:230–232, 2021. doi:10.1016/J.DAM.2020.10.004.

10 Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.

11 Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21(2):358–402, 1996. doi:10.1006/JAGM.1996.0049.

12 Prosenjit Bose, Jurek Czyzowicz, Zhicheng Gao, Pat Morin, and David R. Wood. Simultaneous diagonal flips in plane triangulations. *J. Graph Theory*, 54(4):307–330, 2007. doi:10.1002/JGT.20214.

13 Prosenjit Bose, Dana Jansens, André Van Renssen, Maria Saumell, and Sander Verdonschot. Making triangulations 4-connected using flips. *Comput. Geom.*, 47(2):187–197, 2014. doi:10.1016/J.COMGEO.2012.10.012.

14 Prosenjit Bose and Sander Verdonschot. A history of flips in combinatorial triangulations. In *Abstracts of the XIV Spanish Meeting on Computational Geometry (EGC)*, volume 7579 of *LNCS*, pages 29–44. Springer, 2011. doi:10.1007/978-3-642-34191-5_3.

15 Aaron Büngener and Michael Kaufmann. Improving the crossing lemma by characterizing dense 2-planar and 3-planar graphs. In *Proc. 32nd Symposium on Graph Drawing and Network Visualization (GD)*, volume 320 of *LIPIcs*, pages 29:1–29:22. Schloss Dagstuhl, 2024. doi:10.4230/LIPICS.GD.2024.29.

16 Jean Cardinal, Michael Hoffmann, Vincent Kusters, Csaba D. Tóth, and Manuel Wettstein. Arc diagrams, flip distances, and Hamiltonian triangulations. *Comput. Geom.*, 68:206–225, 2018. doi:10.1016/J.COMGEO.2017.06.001.

17 Ruoxu Cen, Jason Li, and Debmalya Panigrahi. Edge connectivity augmentation in near-linear time. In *Proc. 54th ACM Symposium on Theory of Computing (STOC)*, pages 137–150, 2022. doi:10.1145/3519935.3520038.

18 S. Dhanalakshmi, N. Sadagopan, and V. Manogna. Tri-connectivity augmentation in trees. *Electron. Notes Discret. Math.*, 53:57–72, 2016. doi:10.1016/J.ENDM.2016.05.006.

19 Kapali P. Eswaran and Robert Endre Tarjan. Augmentation problems. *SIAM J. Comput.*, 5(4):653–665, 1976. doi:10.1137/0205044.

20 Sergej Fialko and Petra Mutzel. A new approximation algorithm for the planar augmentation problem. In *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 260–269, 1998. URL: `http://dl.acm.org/citation.cfm?id=314613.314714`.

21 András Frank and Tibor Jordán. Graph connectivity augmentation. In *Handbook of Graph Theory, Combinatorial Optimization, and Algorithms*, chapter 14, pages 313–346. CRC Press, 2015.

22 Greg N. Frederickson and Joseph F. JáJá. Approximation algorithms for several graph augmentation problems. *SIAM J. Comput.*, 10(2):270–283, 1981. `doi:10.1137/0210019`.

23 Jun Fujisawa, Keita Segawa, and Yusuke Suzuki. The matching extendability of optimal 1-planar graphs. *Graphs Comb.*, 34(5):1089–1099, 2018. `doi:10.1007/S00373-018-1932-6`.

24 Jérôme Galtier, Ferran Hurtado, Marc Noy, Stephane Perennes, and Jorge Urrutia. Simultaneous edge flipping in triangulations. *Int. J. Comput. Geom. Appl.*, 13(2):113–133, 2003. `doi:10.1142/S0218195903001098`.

25 Alfredo García, Ferran Hurtado, Clemens Huemer, Javier Tejel, and Pavel Valtr. On triconnected and cubic plane graphs on given point sets. *Comput. Geom.*, 42(9):913–922, 2009. `doi:10.1016/J.COMGEO.2009.03.005`.

26 Alfredo García, Ferran Hurtado, Marc Noy, and Javier Tejel. Augmenting the connectivity of outerplanar graphs. *Algorithmica*, 56(2):160–179, 2010. `doi:10.1007/S00453-008-9167-1`.

27 Carsten Gutwenger, Petra Mutzel, and Bernd Zey. On the hardness and approximability of planar biconnectivity augmentation. In *Proc. 15th Conference on Computing and Combinatorics (COCOON)*, volume 5609 of *LNCS*, pages 249–257. Springer, 2009. `doi:10.1007/978-3-642-02882-3_25`.

28 Carsten Gutwenger, Petra Mutzel, and Bernd Zey. Planar biconnectivity augmentation with fixed embedding. In *Proc. 20th Internatioal Workshop on Combinatorial Algorithms, (IWOCA)*, volume 5874 of *LNCS*, pages 289–300. Springer, 2009. `doi:10.1007/978-3-642-10217-2_29`.

29 Tsan-sheng Hsu. On four-connecting a triconnected graph. *J. Algorithms*, 35(2):202–234, 2000. `doi:10.1006/JAGM.2000.1077`.

30 Dávid Hudák, Tomáš Madaras, and Yusuke Suzuki. On properties of maximal 1-planar graphs. *Discussiones Mathematicae Graph Theory*, 32:737–747, 2012. `doi:10.7151/dmgt.1639`.

31 Ferran Hurtado and Csaba D. Tóth. Plane geometric graph augmentation: a generic perspective. In János Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 327–354. Springer, 2013. `doi:10.1007/978-1-4614-0110-0_17`.

32 Bill Jackson and Tibor Jordán. Independence free graphs and vertex connectivity augmentation. *J. Comb. Theory B*, 94(1):31–77, 2005. `doi:10.1016/J.JCTB.2004.01.004`.

33 Kasper Skov Johansen, Eva Rotenberg, and Carsten Thomassen. Edge-connectivity augmentation of simple graphs. *SIAM J. Discret. Math.*, 39(1):163–169, 2025. `doi:10.1137/23M1574245`.

34 Tibor Jordán. On the optimal vertex-connectivity augmentation. *J. Comb. Theory, Ser. B*, 63(1):8–20, 1995. `doi:10.1006/JCTB.1995.1002`.

35 Goos Kant. Augmenting outerplanar graphs. *J. Algorithms*, 21(1):1–25, 1996. `doi:10.1006/JAGM.1996.0034`.

36 Goos Kant. A more compact visibility representation. *International Journal of Computational Geometry & Applications*, 07(03):197–210, 1997. `doi:10.1142/S0218195997000132`.

37 Goos Kant and Hans L. Bodlaender. Planar graph augmentation problems. In *Proc. 2nd Workshop on Algorithms and Data Structures (WADS)*, volume 519 of *LNCS*, pages 286–298. Springer, 1991. `doi:10.1007/BFB0028270`.

38 Ken-ichi Kawarabayashi and Kenta Ozeki. 4-connected projective-planar graphs are Hamiltonian-connected. *J. Comb. Theory B*, 112:36–69, 2015. `doi:10.1016/J.JCTB.2014.11.006`.

39 Ton Kloks. *Treewidth: Computations and Approximations*, volume 842 of *LNCS*. Springer, 1994. `doi:10.1007/BFB0045375`.

40 Frank Thomson Leighton. *Complexity Issues in VLSI: Optimal Layouts for the Shuffle-Exchange Graph and Other Networks*. MIT Press, 1983. URL: https://mitpress.mit.edu/9780262621786/complexity-issues-in-vlsi/.

41 David Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982. doi:10.1137/0211025.

42 Ryuichi Mori, Atsuhiro Nakamoto, and Katsuhiro Ota. Diagonal flips in hamiltonian triangulations on the sphere. *Graphs Comb.*, 19(3):413–418, 2003. doi:10.1007/S00373-002-0508-6.

43 Hiroshi Nagamochi and Peter Eades. Edge-splitting and edge-connectivity augmentation in planar graphs. In *Proc. 6th Integer Programming and Combinatorial Optimization (IPCO)*, volume 1412 of *LNCS*, pages 96–111. Springer, 1998. doi:10.1007/3-540-69346-7_8.

44 Kenta Ozeki and Carol T. Zamfirescu. Every 4-connected graph with crossing number 2 is Hamiltonian. *SIAM J. Discret. Math.*, 32(4):2783–2794, 2018. doi:10.1137/17M1138443.

45 János Pach and Géza Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17:427–439, 1997. doi:10.1007/BF01215922.

46 David Rappaport. Computing simple circuits from a set of line segments is NP-complete. *SIAM J. Comput.*, 18(6):1128–1139, 1989. doi:10.1137/0218075.

47 Ignaz Rutter and Alexander Wolff. Augmenting the connectivity of planar and geometric graphs. *J. Graph Algorithms Appl.*, 16(2):599–628, 2012. doi:10.7155/JGAA.00275.

48 Yusuke Suzuki. Re-embeddings of maximum 1-planar graphs. *SIAM J. Discret. Math.*, 24(4):1527–1540, 2010. doi:10.1137/090746835.

49 Robin Thomas and Xingxing Yu. 4-connected projective-planar graphs are Hamiltonian. *J. Comb. Theory B*, 62(1):114–132, 1994. doi:10.1006/JCTB.1994.1058.

50 Csaba D. Tóth. Connectivity augmentation in planar straight line graphs. *European J. Combinatorics*, 33:408–425, 2012. doi:10.1016/j.ejc.2011.09.002.

51 Vera Traub and Rico Zenklusen. Local search for weighted tree augmentation and Steiner tree. In *Proc. 33rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3253–3272, 2022. doi:10.1137/1.9781611977073.128.

52 William T. Tutte. A theorem on planar graphs. *Trans. Amer. Math. Soc.*, 82:99–116, 1956. doi:10.1090/s0002-9947-1956-0081471-8.

53 László A. Végh. Augmenting undirected node-connectivity by one. *SIAM J. Discret. Math.*, 25(2):695–718, 2011. doi:10.1137/100787507.

54 Toshimasa Watanabe and Akira Nakamura. Edge-connectivity augmentation problems. *J. Comput. Syst. Sci.*, 35(1):96–144, 1987. doi:10.1016/0022-0000(87)90038-9.

55 Toshimasa Watanabe and Akira Nakamura. A minimum 3-connectivity augmentation of a graph. *J. Comput. Syst. Sci.*, 46(1):91–127, 1993. doi:10.1016/0022-0000(93)90050-7.

String Graph Obstacles of High Girth and of Bounded Degree

Maria Chudnovsky
Mathematics Department, Princeton University, NJ, USA

David Eppstein
Computer Science Department, University of California, Irvine, CA, USA

David Fischer
Mathematics Department, Princeton University, NJ, USA

Abstract

A string graph is the intersection graph of curves in the plane. Kratochvíl previously showed the existence of infinitely many obstacles: graphs that are not string graphs but for which any edge contraction or vertex deletion produces a string graph. Kratochvíl's obstacles contain arbitrarily large cliques, so they have girth three and unbounded degree. We extend this line of working by studying obstacles among graphs of restricted girth and/or degree. We construct an infinite family of obstacles of girth four; in addition, our construction is $K_{2,3}$-subgraph-free and near-planar (planar plus one edge). Furthermore, we prove that there is a subcubic obstacle of girth three, and that there are no subcubic obstacles of high girth. We characterize the subcubic string graphs as having a matching whose contraction yields a planar graph, and based on this characterization we find a linear-time algorithm for recognizing subcubic string graphs of bounded treewidth.

2012 ACM Subject Classification Mathematics of computing → Graphs and surfaces

Keywords and phrases string graphs, induced minors, forbidden minors, sparsity, triangle-free graphs, near-planar graphs

Digital Object Identifier 10.4230/LIPIcs.GD.2025.24

Funding *Maria Chudnovsky*: Supported in part by NSF Grants DMS-2348219 and CCF-2505100.
David Eppstein: Research supported in part by NSF grant CCF-2212129.

1 Introduction

String graphs, the intersection graphs of curves in the plane [13], developed out of the study of the patterns of mutations in DNA sequences [5], and of the crossing patterns of wires in circuit designs [26]. Beyond the two-dimensional nature of their definitions, string graphs and planar graphs are closely related, although with several important differences. For example, the two classes exhibit similar (but not identical) closure properties.

▶ **Lemma 1.** *The class of planar graphs is closed under taking minors, and the class of string graphs is closed under taking induced minors [20]. Explicitly:*

- *If G is a planar graph, then any graph obtained from G by a sequence of vertex deletions, edge deletions, and edge contractions is also a planar graph.*
- *If G is a string graph, then any graph obtained from G by a sequence of vertex deletions and edge contractions (but not edge deletions) is also a string graph.*

In general, the class of string graphs is larger than the class of planar graphs, but there is a class of non-string graphs that arises from the class of non-planar graphs. This relationship is formalized in the following lemma, where for a graph G, we denote by $\mathrm{sub}_1(\mathrm{G})$ the graph obtained from G by subdividing each edge once.


© Maria Chudnovsky, David Eppstein, and David Fischer;
licensed under Creative Commons License CC-BY 4.0

▶ **Lemma 2.** *Let G be a graph, and let H denote any graph obtained from G by subdividing each edge at least once, such as* $\mathrm{sub}_1(\mathrm{G})$*. Then, the following hold:*

- *If G is a planar graph, then H is a string graph.*
- *If G is a non-planar graph, then H is a non-string graph [13, 26].*

Famously, the class of planar graphs is characterized by two forbidden minors:

▶ **Theorem 3** (Wagner [27])**.** *A graph G is non-planar if and only if G contains K_5 or $K_{3,3}$ as a minor.*

As a consequence, planar graphs can be recognized in polynomial time [24].

Due to the relationship between planar graphs and string graphs described by Lemma 2, a natural question that arises is whether the non-string graphs are exactly those that are obtained by subdividing non-planar graphs. Given Theorem 3, this idea can be formulated as follows.

▶ **Premise 4.** *A graph G is a non-string graph if and only if G contains* $\mathrm{sub}_1(\mathrm{K}_5)$ *or* $\mathrm{sub}_1(\mathrm{K}_{3,3})$ *as an induced minor.*

If Premise 4 were true, it would provide a full structural characterization of the string graphs, allowing for many results from the study of planar graphs to carry into string graphs. However, it is known to be false in general, with string graphs exhibiting much more complex behavior than planar graphs; there are infinitely many *string graph obstacles* [18], i.e. graphs that are not string graphs but for which any proper induced minor is a string graph. Furthermore recognition of string graphs is NP-complete [19, 25].

Given the extensive literature in the study of planar graphs, it is of interest to understand for which subclasses of string graphs the statement of Premise 4 holds; or, more generally, for which subclasses there are only finitely many obstructions. The infinite family of obstacles constructed in [18] has unbounded clique size, so there is no non-trivial lower bound on the girth of the family, and there is no upper bound on the maximum degree. On the other hand, the non-string graphs $\mathrm{sub}_1(\mathrm{K}_5)$ and $\mathrm{sub}_1(\mathrm{K}_{3,3})$ are both triangle-free and have maximum degree five and three, respectively. Thus, two natural starting points to search for regimes under which Premise 4 holds are the restrictions of string graphs to high girth and to bounded degree. These regimes are the main subject of this paper–both in the context of Premise 4, and more generally in determining whether these classes contain finitely many or infinitely many string graph obstacles.

In Section 3, we study the regime of graphs with girth at least four, i.e. triangle-free graphs, proving the following.

▶ **Theorem 5.** *There exist infinitely many string graph obstacles of girth four.*

We note that in addition to being triangle-free, our construction is also $K_{2,3}$-free and near-planar (planar plus one edge – for more on near-planar graphs see [6]). However, it does not have bounded degree.

Next, in Section 4 we study the graphs of maximum degree three, the smallest non-trivial upper bound on degree. We prove that even in this regime, Premise 4 does not hold. However, we also show that obstacles in this regime cannot have arbitrarily large girth.

▶ **Theorem 6.** *There exist string graph obstacles of maximum degree three apart from* $\mathrm{sub}_1(\mathrm{K}_{3,3})$*. All such obstacles have girth smaller than* 30*.*

To prove Theorem 6, we obtain a novel characterization of the subcubic string graphs:

▶ **Theorem 7.** *Let G be a graph with maximum degree three. Then G is a string graph if and only if there exists a matching M of G such that the graph obtained from G by contracting M is planar.*

Finally, in Section 5, we finish by proving an algorithmic corollary of Theorem 7:

▶ **Theorem 8.** *The problem of deciding whether a graph with maximum degree three is a string graph is fixed-parameter tractable in treewidth. For any graph class of maximum degree 3 and bounded treewidth, the problem is decidable in linear time.*

We note that recognition of string graphs of maximum degree four is an NP-complete problem [19, 23], whereas the complexity is open for graphs of maximum degree three. It is known that it is NP-complete to check whether a graph has a matching that planarizes the graph when contracted [1, 2], but this is without any restriction on the maximum degree of the input graph.

2 Preliminaries

By $\mathbb{N}$ we denote the set of positive integers. We define a *string representation* of a string graph to be a system of curves, called *strings*, associated with the vertices of the graph, that have the given graph as their intersection graph. Rather than considering arbitrary systems of curves in the plane, which may cross each other infinitely many times or even cover nonzero area, it is convenient to consider certain more well-behaved but still fully general string representations. A *Jordan arc* is a curve ambient isotopic to a line segment in the plane. As a degenerate case we also consider a single point to be a Jordan arc. We define a *proper string representation* to be a string representation in which each string is a Jordan arc, each two strings have finitely many points of intersection, at most two strings intersect at any point, and each intersection point is either a crossing of two strings or an endpoint of at least one of the intersecting strings. The proof of the following fact is straightforward and we omit it.

▶ **Lemma 9.** *Every string graph has a proper string representation.*

Each subset of the strings of a proper string representation defines a proper string representation of an induced subgraph. That is not true for the next simplifying assumption to make. A *trimmed string representation* is a proper string representation in which each string is a Jordan arc, and in which it is not possible to replace any string by a proper subset of itself and obtain a string representation of the same graph. That is, each string is minimal among arcs having crossings with the same set of strings. A trimmed string representation may be obtained from a proper string representation by replacing each string by a minimal arc, one at a time.

▶ **Observation 10.** *In a trimmed string representation, each string endpoint is an intersection point of two strings. Two strings can intersect either in an endpoint of one or both strings, or in one or more crossing points, but not both.*

Proof. If two strings had an intersection point at an endpoint, and also another intersection point at a crossing, they would not be minimal for the set of strings that they cross, because the string that ends at an intersection point could be shortened without changing the set of strings that it crosses. ◀

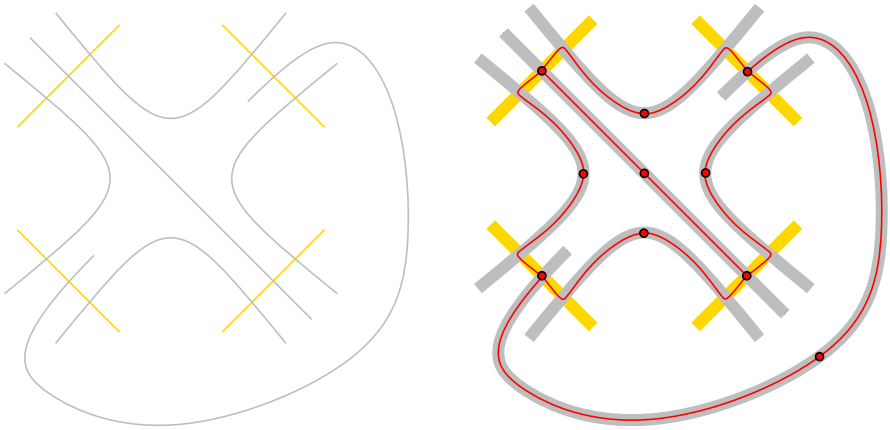


Figure 1 An example of Observation 11. On the left is a string representation of $\mathrm{sub}_1(K_4)$, and on the right is a planar representation within (the trimmed part of) a thickening of the string representation.

It is known that, when a graph is formed from another graph by subdividing each edge into a path of two or more edges, the subdivided graph is a string graph if and only if the original graph is planar [26]. This idea can be formalized as the following observation:

▶ **Observation 11.** *If a string graph has the property that every edge has a degree-two endpoint, then it is a planar graph, and any trimmed string representation forms a contact representation by a system of Jordan arcs. If these arcs are thickened to topological disks, then the union of the resulting system of disks contains a planar drawing of the graph in which each vertex is represented by a point within its disk and each edge is represented by a curve within the union of two disks connecting two of these representative points.*

Proof. No degree-two vertex can have a string with a crossing, because its two endpoints represent its only two adjacencies. If every edge has a degree-two endpoint, there can be no crossings, because each two intersecting strings include one that can have no crossings. ◀

Here the planarity of the resulting graph requires that we are considering contact graphs of curves for which at most two can intersect in any point. Contact systems of curves without a restriction to pairwise intersections may be nonplanar [17].

3 Triangle-free Obstacles

Our family of triangle-free obstacles is built from a family of vertex-colored planar graphs, an example of which is depicted in Figure 2. This graph has the overall form of a cycle, in which components in the form of two different types of subgraph are connected by pairs of *terminal vertices*, colored white. Each component has two terminal vertices, each one shared with one of its two neighboring components in the cycle. The two types of components are:

- Any number of *links*, four-vertex cycles consisting of two non-adjacent white terminals, a red vertex, and a blue vertex.
- The *pendant*, a tree formed as the 1-subdivision of the claw $K_{1,3}$. All but one vertex are white; the remaining vertex, a leaf of the tree, is colored both red and blue, and we call it the *jewel*. The two terminals are the other two leaves of the tree.

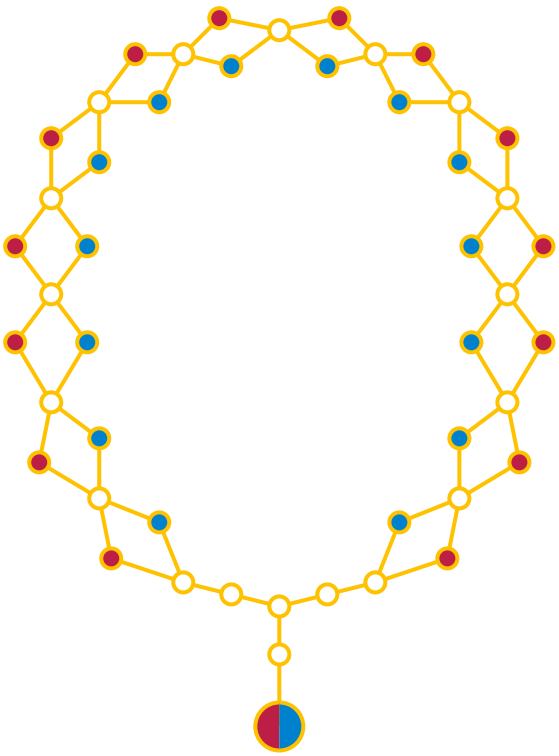


Figure 2 The necklace graph N_{14}.

For each $i \in \mathbb{N}$, we define the necklace N_i to be a necklace whose cycle of components consists of i links and one pendant. For instance the necklace in the figure can be denoted in this way as N_{14}. We observe that all planar drawings of this graph are topologically equivalent if neither the vertex colors nor the distinction between bounded and unbounded faces is considered. Each link bounds a quadrilateral face. There are two more faces, one polygonal face with $4 + 2i$ sides not containing the jewel, and a face with $8 + 2i$ sides (some of them repeated) containing the jewel. By convention we call these last two faces the *inner face* and the *outer face*, respectively, matching the way that they are drawn in the figure. Alternative planar embeddings may rearrange the colors of some of the vertices on these faces but not their topological structure.

We are now ready to describe our family of triangle-free obstacles. They are the graphs $\hat{N}_i$, obtained from a necklace N_i by adding two more vertices: a red apex and a blue apex. The red apex is adjacent to all the red vertices of N_i, including the jewel. The blue apex is adjacent to all the blue vertices of N_i, including the jewel. These are the only additional adjacencies.

▶ **Observation 12.** *For each $i \in \mathbb{N}$, $\hat{N}_i$ is triangle-free, $K_{2,3}$-subgraph-free, and near-planar.*

Proof. Each component of N_i is triangle-free, and the inner and outer face are too long to have a triangle, so N_i is triangle-free. No two red vertices are adjacent, so there is no triangle involving the red apex, and symmetrically no two blue vertices are adjacent, so there is no triangle involving the blue apex. At most two blue vertices of N_i share a common white neighbor, so no $K_{2,3}$ subgraph can include the blue apex, and symmetrically with the red apex, so any $K_{2,3}$ subgraph of $\hat{N}_i$ can only exist within N_i itself. However, the only four-cycles in N_i are the links, which do not form $K_{2,3}$ subgraphs.

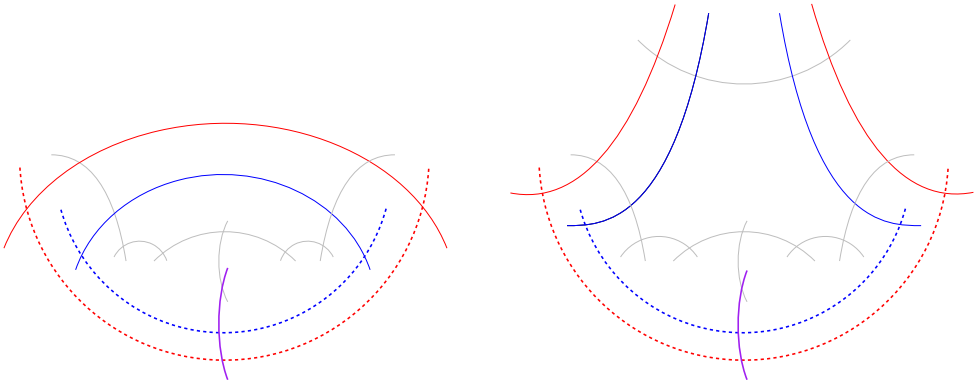


Figure 3 String representations of $\hat{N}_1$ (left) and $\hat{N}_2$ (right). The light gray strings represent white vertices, and the solid red and blue strings represent red and blue vertices, respectively. The dashed red and blue strings represent the red and blue apexes, respectively, and the purple string represents the jewel.

When the necklace is drawn as in the figure, with all blue vertices on the inner face and all red vertices on the outer face, the blue apex can be placed in the inner face and connected planarly to all blue vertices except the jewel. The red apex can be placed in the outer face and connected planarly to all red vertices including the jewel. Only one edge, from the blue apex to the jewel, is missing. ◀

▶ **Lemma 13.** *For $i \in \mathbb{N}$, $\hat{N}_i$ is a string graph if and only if $i \in \{1, 2\}$.*

Proof. String representations of $\hat{N}_1$ and $\hat{N}_2$ are shown in Figure 3.

We now assume that $i \geq 3$. Suppose that S is a string representation of $\hat{N}_i$. The induced subgraph N_i obeys Observation 11; obtain a trimmed representation of N_i by removing the strings for the two apexes and trimming, and let D be a planar drawing of N_i, within a thickening of this trimmed representation, following Observation 11.

Observe that N_i contains a unique cycle C_R comprised only of red and white vertices, and similarly N_i contains a unique cycle C_B comprised only of blue and white vertices. Then any cycle in D separates the plane into two sides, restricting any string in S that does not have a neighbor in the cycle to only one of these two sides. Thus, in D, C_R separates the plane into two faces, which we denote by $F^+_{C_R}$ and $F^-_{C_R}$. Similarly, C_B separates the plane into two faces, which we denote by $F^+_{C_B}$ and $F^-_{C_B}$. Since the jewel does not have a neighbor in C_R, its string in S cannot pass through both $F^+_{C_R}$ and $F^-_{C_R}$; without loss of generality, we assume that it only passes through $F^+_{C_R}$. Similarly, we assume that the string of the jewel only passes through $F^+_{C_B}$ and does not pass through $F^-_{C_B}$. Thus, the jewel string lies entirely in $F^+_{C_R} \cap F^+_{C_B}$.

We observe that the red apex only leaves $F^+_{C_R} \cap F^+_{C_B}$ in order to enter sections of the plane that are bound by 4-cycles of D, and each such cycle C consists of one red vertex, one blue vertex, and two white vertices. In particular, as the red vertex is the only vertex in C that is adjacent to the red apex, the red apex must enter and exit the section bound by C through the red vertex. It is easy to see that if this is the case, then S can be modified so that the red apex string does not enter the section bound by C, by extending the red vertex's string to intersect the red apex string within $F^+_{C_R} \cap F^+_{C_B}$. Thus, without loss of generality, we may assume that the red apex string lies entirely in $F^+_{C_R} \cap F^+_{C_B}$, and symmetrically that the blue apex string also lies entirely in $F^+_{C_R} \cap F^+_{C_B}$. From here, we deduce that there exists a string

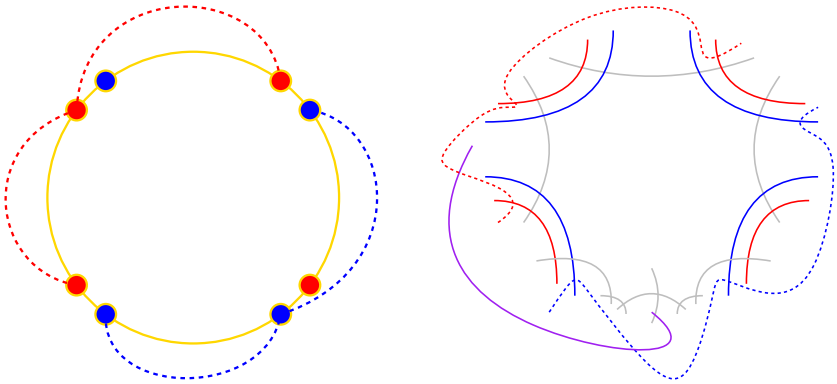


Figure 4 An example demonstrating the idea of the reduction in the proof of Lemma 13. On the left is an attempt at creating a satisfying H_4 as described in the proof; the only condition violated is Condition 5, that each of the monochromatic induced subgraphs (i.e. the subgraph on all red vertices and edges, and the one on all blue vertices and edges) is connected, as each is the disjoint union of a path on 3 vertices and an isolated vertex. However, any attempt to add edges to satisfy Condition 5 causes the graph to violate Condition 6. On the right is the corresponding attempt to draw a string representation of $\hat{N}_4$. Each apex string intersects (apart from the jewel) three of the four strings that it needs to in order to achieve a string representation of $\hat{N}_4$, and these three strings correspond to the path on three vertices in the monochromatic subgraph of the left graph, while the string not intersected by the apex represents the isolated vertex in the monochromatic subgraph of the left graph.

representation of $\hat{N}_i$ if and only if there exists a graph H_i and a coloring $c : V(H_i) \to \{R, B\}$ such that

1. $V(H_i) = \{v_1^1, v_1^2, v_2^1, v_2^2, \ldots, v_n^1, v_n^2\}$, and
2. $v_1^1 v_1^2 \ldots v_n^1 v_n^2 v_1^1$ is a cycle (which we will denote by C_i), and
3. $c(v_j^1) \neq c(v_j^2)$ for all $1 \leq j \leq n$, and
4. $c(x) = c(y)$ for all $xy \in E(H_i) \setminus E(C_i)$, and
5. $H_i[R_i] \setminus E(C_i)$ and $H_i[B_i] \setminus E(C_i)$ are both connected, where $R_i = \{v \in V(H_i) : c(v) = R\}$ $B_i = \{v \in V(H_i) : c(v) = B\}$, and
6. there exists a planar drawing of H_i such that all edges lie entirely in the exterior of C_i.

The intuition behind the equivalence between these two problems is that, if a string representation of $\hat{N}_i$ exists, then the string representation gives rise to an H_i as above, with the apex strings defining the non-cycle edges of H_i. Conversely, an H_i as described above gives rise to a string representation of $\hat{N}_i$, with the non-cycle edges of each color defining that color's respective apex string. This relationship is shown in Figure 4.

Regardless of the choice of H_i and c, there exist vertices $r_1, r_2, b_1, b_2 \in V(H_i)$ with $c(r_1) = c(r_2) = R$ and $c(b_1) = c(b_2) = B$ such that $r_1 r_2, b_1 b_2 \in E(H_i)$ and the cyclic order of these vertices on C_i is r_1, b_1, r_2, b_2. It follows that in any planar drawing of H_i, it is impossible for both $r_1 r_2$ and $b_1 b_2$ to lie entirely in the exterior of C_i without crossing. Thus, $\hat{N}_i$ is not a string graph when $i \geq 3$. ◀

▶ **Lemma 14.** *For each $i \in \mathbb{N}$, every proper induced minor of $\hat{N}_i$ is a string graph.*

Proof. We need only consider induced minors that perform a single vertex deletion or a single edge contraction in $\hat{N}_i$. Let e be the edge connecting the jewel to the blue apex; we will start with the planar drawing of $\hat{N}_i - e$ obtained in Observation 12, and its string representation

obtained from this drawing by tracing a string around each vertex and its incident half-edges. We will show that in each possible vertex deletion $\hat{N}_i - v$ or edge contraction $\hat{N}_i/f$, it is possible to modify the string representation of $\hat{N}_i - e$ to obtain a string representation of $\hat{N}_i - v$ or $\hat{N}_i/f$.

For vertex deletions $\hat{N}_i - v$, we have the following cases:

- If v is a white vertex belonging to both the inner and outer face of N_i, then deleting it and its string produces a string representation of $\hat{N}_i - v$ in which the inner and outer faces are no longer separated from each other. We can extend the string for the jewel to cross through the string for the red apex and through this gap, from the outer face to the inner face, where it can cross the string for the blue apex without being obstructed by any other string.
- If v is a red vertex, then deleting it and its string produces a string representation of $\hat{N}_i - v$ that breaks the red–white cycle which blocked the blue apex from passing through both the inner and outer faces. The string for the blue apex can be extended to cross through the blue vertex in the same link as v, into the outer face of $N_i - v$. It cannot cross the string for the red apex, but the string for the jewel can also be extended to cross the string for the red apex and meet the extended string for the blue vertex, adding the missing edge e to the string representation of $\hat{N}_i - v$.
- If v is a blue vertex of N_i, then deleting it is symmetric to deleting a red vertex under a symmetry of $\hat{N}_i$ that swaps the two colors red and blue.
- If v is the blue apex or the jewel, then deleting it obviates the need to represent the missing edge e. The case where v is the red apex is symmetric.
- The only remaining vertex that could be deleted is the white vertex neighboring the jewel. If this vertex is deleted, the string for the jewel can be placed within any link. The strings for each apex can be extended through the vertex of the link of the same color, into the link, where they can separately intersect the string for the jewel.

For edge contractions $\hat{N}_i/f$, we have the following cases:

- If f is a blue–white edge of a link, contracting it produces a contracted vertex that is blue (incident to the blue apex) and that belongs to both the inner and outer face. The string for the blue apex can cross through the string for this contracted vertex, into the outer face, where it can be met by an extension of the string for the jewel, much like the case of deleting a red vertex. Contracting a red–white edge of a link is symmetric.
- If f is an edge connecting the blue apex to a blue vertex, contracting it produces a graph in which the blue apex has a white neighbor on both the inner and outer face. The string for the blue apex can extend through the string for this neighbor, into the outer face, where it can be met by an extension of the string for the jewel. Contracting an edge connecting the red apex to a red vertex is symmetric.
- If f is an edge on the four-edge path connecting the two terminals of the pendant, then contracting it produces a graph in which the degree-3 vertex of the pendant is adjacent to one terminal. The string for this degree-3 vertex can cross through the neighboring terminal, into its adjacent link. This allows the jewel and its neighbor to be placed within this link, instead of in the inner or outer face. The strings for each apex can cross through the strings for the red and blue vertices of this link, allowing the apex strings to reach the string for the jewel.
- If f is one of the two remaining edges of the pendant, then contracting it makes the jewel adjacent to the white degree-3 vertex, which belongs to both the inner and outer face. The string for the jewel can cross through the string for this white neighbor, into the inner face, where it can be intersected by the string for the blue apex.

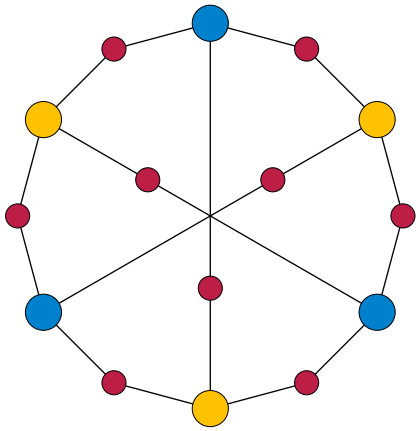


Figure 5 The 1-subdivision of $K_{3,3}$, the only previously known subcubic obstacle to string graphs.

- If f is the edge connecting the red apex to the jewel, then after it is contracted and the two strings for the apex and jewel are combined into a single string, this combined string still needs to intersect the string for the blue apex. It can do so by crossing the string for any red vertex into any link, where it can be met by an extension of the blue apex string that crosses the string for the blue vertex of the same link. The case where f connects the blue apex to the jewel is symmetric.

Each case produces a valid string representation, from which the result follows. ◀

▶ **Theorem 15.** *There are infinitely many triangle-free, $K_{2,3}$-subgraph-free, near-planar graphs that are minimal as non-string graphs in the induced minor order.*

Proof. The graphs $\hat{N}_i$ for $i \geq 3$ are non-isomorphic (they have different numbers of vertices), triangle-free, $K_{2,3}$-free, and near-planar by Observation 12, non-string graphs by Lemma 13, and minimal with this property for the induced minor order by Lemma 14. ◀

4 Subcubic Obstacles

A graph G is *subcubic* if its maximum degree is at most three, and a *cubic matching of* G is a matching M using only cubic edges (edges that have two degree-3 endpoints). The only previously known subcubic obstacle to string graphs was the 1-subdivision of $K_{3,3}$ (Figure 5), which has girth eight. In this section we present new subcubic obstacles to string graphs including one that contains a triangle. Additionally, we limit the search for other subcubic obstacles by proving that they do not have arbitrarily large girth.

We do not know whether the string graphs have a finite or infinite number of subcubic obstacles. It is known that there exist infinite antichains of bounded-degree graphs in the induced minor order [21], but even this much appears not to have been known for maximum degree three. In Appendix B we describe an infinite antichain of cubic toroidal graphs, and relate them to string graphs, however without determining whether they can be used to construct additional subcubic obstacles.

We base many of our results in this section on the following characterization of subcubic string graphs in terms of contractions of matchings.

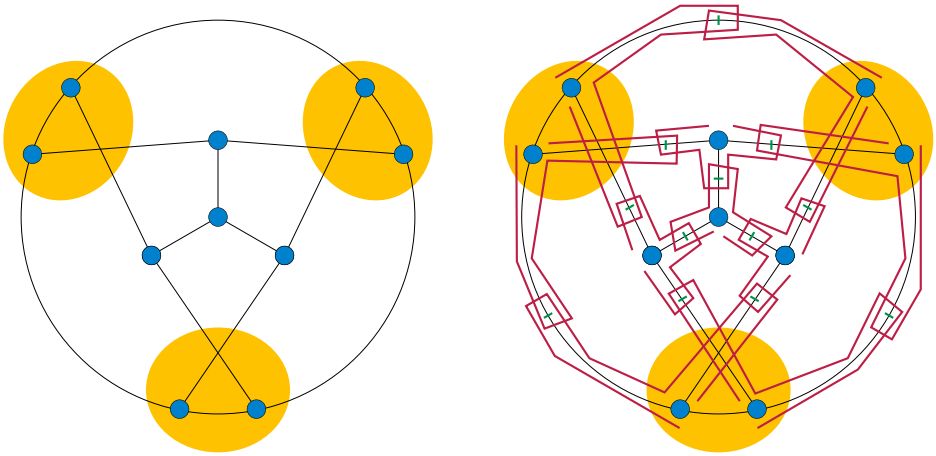


Figure 6 Left: A three-edge matching (shaded ovals) in the Petersen graph whose contraction produces a planar graph (all crossings are inside ovals). Right: A string graph representation of the Petersen graph constructed following the proof of Theorem 16. The green marks subdivide each unmatched edge into half-edges, which the red strings trace around.

▶ **Theorem 16.** *If a graph G has a matching M such that the contraction of the matched edges, G/M, is planar, then G is a string graph. Every subcubic string graph has such a matching that is cubic.*

Proof. For the first claim of the theorem, suppose that G has a matching M for which G/M is planar; we show that G is a string graph. Find a planar drawing of G/M, and expand each vertex in the drawing, coming from a contracted edge in M, from a point into a topological disk, disjoint from the remaining components of the drawing. Figure 6 depicts an example, with these expanded disks shown in yellow. Let D be the resulting vertex-expanded drawing. Place a point arbitrarily on each edge of D, disjoint from the endpoints of the edge, separating the edge into two topological half-edges. Represent G as a string graph in which each unmatched vertex is represented by a string that traces around the vertex and its incident half-edges in D. For each vertex v in a matched edge vw, represent v as a string which stays within the expanded disk representing the contracted vertex for vw, crossing the string for w, and extends beyond this expanded disk only to trace around each half-edge connecting this contracted vertex to a neighbor of v in G. The resulting system of strings intersect within the expanded disks for each contracted edge, and near the subdivision point along each uncontracted edge representing an adjacency in G/M; there are no other crossings. Thus, these strings form a string representation of G.

For the second claim of the theorem, suppose that G is a subcubic string graph. We must show that there exists a cubic matching M such that G/M is planar. Consider any trimmed representation of G. Each string in this representation, representing a vertex v, must touch strings for two distinct neighbors of v at its two endpoints; it can intersect at most one other string than these two neighbors, a third neighbor w of v, at proper crossings. If the string for v crosses the string for w in this way, for the same reason, w can have no other crossings than with v. Thus, the pairing of vertices of G whose strings cross each other is a matching M. When the two strings for each matched pair are replaced by their union, a connected subset of the plane, the result is a contact representation of G/M, by sets (curves and unions of two curves) for which at most two sets intersect at any point, which is therefore planar as

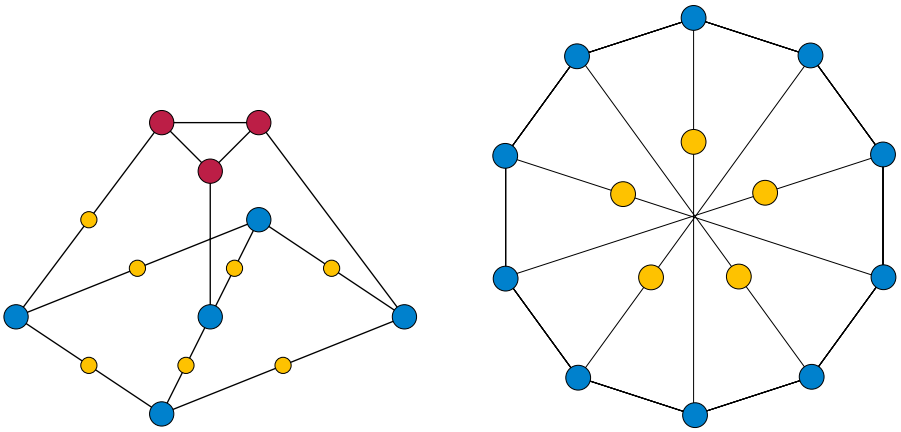


Figure 7 Subcubic obstacles KT_3 (left) and SM_{10} (right).

in Observation 11. Furthermore, two strings can only intersect at a proper crossing if both vertices have degree three. Thus, every edge of M is adjacent to two vertices of degree three, so M is a cubic matching. ◀

4.1 New Subcubic Obstacles

As an example, this characterization can be used to show that the 1-subdivision of $K_{3,3}$ (Figure 5) is a minimal non-string graph. It is not a string graph by Observation 11. If any vertex is deleted, the remaining subgraph becomes planar, because it forms a subdivision of a vertex deletion or edge deletion of $K_{3,3}$. And if any of its edges is contracted, replacing a two-edge subdivided path uvw by the single edge uw, then the matching consisting only of edge uw meets the conditions of Theorem 16: contracting it produces a subdivision of the planar wheel graph obtained by contracting one edge of $K_{3,3}$. Thus, all induced minors of the 1-subdivision of $K_{3,3}$ are string graphs.

We use this characterization to show the existence of several novel subcubic obstacles, depicted in Figure 7. The first is KT_3, which is constructed from the 1-subdivision of $K_{3,3}$ by adding an edge between two subdivision vertices at distance two from each other.

▶ **Proposition 17.** *KT_3 is a minimal non-string graph.*

We note that additional obstructions can be obtained in a similar fashion by replacing multiple vertices of the 1-subdivision of $K_{3,3}$ with triangles (rather than just one).

Our second subcubic obstacle, SM_{10}, is not obtained from $K_{3,3}$, and its existence demonstrates that the 1-subdivision of $K_{3,3}$ is not the only triangle-free subcubic obstacle. It may be formed from the 10-vertex Möbius ladder [16] by subdividing the five "rungs" of the ladder; it consists of a 10-cycle of degree-3 vertices, together with five two-edge paths connecting opposite pairs of vertices of the 10-cycle.

▶ **Proposition 18.** *SM_{10} is a minimal non-string graph.*

The proofs of Proposition 17 and Proposition 18 are direct applications of Theorem 16; for the full proofs see Appendix A. For a partial proof that the 14-vertex Heawood graph is also a minimal non-string graph, see Appendix B.

4.2 Girth of Subcubic Obstacles

Since KT_3 contains a triangle, there is no nontrivial lower bound on the girth of subcubic obstacles. On the other hand, in what follows we show an upper bound. We utilize the following well-known bound on the edge density of planar graphs with high girth:

▶ **Lemma 19.** *Let G be a connected planar graph with at least 3 vertices and with girth at least g. Put $n = |V(G)|$ and $m = |E(G)|$. Then*

$$\frac{m}{n} \leq \frac{g}{g-2}.$$

We show that the properties of being a subcubic obstacle imply that the edge density of any such obstacle is bounded above c for a constant $c > 1$, then use Lemma 19 to obtain an upper bound on the edge density of the obstacle that is smaller than c if its girth is too large. The details are presented below.

▶ **Theorem 20.** *Let G be a subcubic non-string graph such that every proper induced minor of G is a string graph. Then G has girth smaller than 30.*

Proof. We begin by noting several properties of G that follow from the fact that G is a subcubic obstacle. Firstly, G is connected, and $G \setminus v$ is connected for any $v \in V(G)$ of degree 2. Additionally, G does not contain any vertex v with $\deg(v) \in \{0, 1\}$. Further, G does not contain two adjacent vertices u, v of degree two.

Since G has no vertices of degree 0 or 1, G is not a forest, and thus has finite girth. Suppose for contradiction that G has girth at least 30; then $n := |V(G)| \geq 30$. Since every $e \in E(G)$ has at least one endpoint of degree 3, G has density at least $\frac{6}{5}$, which is the density of the graph obtained from a cubic graph by subdividing each edge once. If G is cubic, let $v \in V(G)$ be any vertex such that $G \setminus v$ is connected, and otherwise let $v \in V(G)$ be a vertex of degree 2. If G is cubic then it is easily verified that $G \setminus v$ has density at least $\frac{6}{5}$. If G is not cubic, then $G \setminus v$ has $n - 1$ vertices and at least $\frac{6}{5}n - 2$ edges. Thus the density of $G \setminus v$ is at least $\frac{\frac{6}{5}(30)-2}{30-1} = \frac{34}{29}$.

Since $G \setminus v$ is a proper induced minor of G, it is a string graph, and so by Theorem 16 there is a matching M of $G \setminus v$ such that $(G \setminus v)/M$ is planar. Since $G \setminus v$ has girth bigger than 3, contracting M does not create any parallel edges, and so $(G \setminus v)/M$ has $n - |M|$ vertices and at least $\frac{34}{29}n - |M|$ edges, thus the density of G is at least $\frac{\frac{34}{29}n-|M|}{n-|M|} \geq \frac{34}{29}$. $(G \setminus v)/M$ has girth at least 15, since $G \setminus v$ has girth at least 30 and at most half of the edges in any cycle of $G \setminus v$ can be contained in M. Since $G \setminus v$ is connected, $(G \setminus v)/M$ is also connected. As $(G \setminus v)$ is planar, by Lemma 19 we have that the density of $(G \setminus v)/M$ is at most $\frac{15}{13}$, which is smaller than $\frac{34}{29}$ and thus a contradiction. ◀

5 Subcubic Recognition

Courcelle's theorem [11] is a powerful algorithmic metatheorem for converting logical statements about graphs into algorithms. It concerns the *monadic second-order logic of graphs* MSO_2, which allows quantified formulas over vertices, edges, vertex sets, and edge sets, with predicates for testing set membership and vertex–edge incidence. According to Courcelle's theorem, the problem of testing whether a graph models a formula of this type can be solved in time that is linear in the number of vertices in the graph, multiplied by a (non-elementary but computable) function of the formula length and the treewidth of the graph. Thus, for any fixed formula, recognizing the graphs that model the formula is fixed-parameter tractable

in the treewidth of the graph. In order to design an algorithm of this type for a graph recognition problem, it is necessary only to find a logical formula characterizing the graphs to be recognized. This methodology has been frequently used for problems in graph drawing and topological graph theory [4, 7, 8, 14, 22], including by Courcelle himself [9, 10].

▶ **Observation 21.** *The characterization of subcubic string graphs of Theorem 16 can be expressed as a formula in* MSO_2.

Proof. At a high level, we express in logic the existence of a set of edges M such that contracting M produces a graph that has neither K_5 nor $K_{3,3}$ as a minor. The existence of a K_5 minor, or a $K_{3,3}$ minor, in the contracted graph, can be expressed in MSO_2 logic as the existence of five or six vertex sets of the original uncontracted graph, corresponding to the vertices of the minor, with the following properties:

- Any two of these vertex sets are disjoint: there does not exist a vertex belonging to any two of them.
- Each of these five or six vertex sets forms a connected subgraph of the given graph. That is, for each such set S, there is no cut $(S \cap T, S \setminus T)$ of the subgraph induced by S. Here, the existence of a cut may be expressed logically as the existence of a vertex set T for which $S \cap T$ and $S \setminus T$ are nonempty and for which there does not exist an edge with one endpoint in $S \cap T$ and the other endpoint in $S \setminus T$.
- For each edge (u, v) in the minor, represented by the vertex sets U and V, there exists an edge in the given graph whose two endpoints belong to U and V.
- The minor respects the contraction specified by M: for each edge e in M, it is not the case that one of the endpoints of e belongs to one of the five or six vertex sets of the minor but the other endpoint does not belong to the same set.

Each of these properties is straightforward to formalize in MSO_2 logic. ◀

▶ **Corollary 22.** *Subcubic string graphs can be recognized in time that is fixed-parameter tractable with respect to treewidth. Within any subcubic graph family $\mathcal{F}$ of bounded treewidth, string graphs can be recognized in time that is linear in the representation of the input graph.*

References

1 James Abello, Pavel Klavík, Jan Kratochvíl, and Tomáš Vyskočil. Msol restricted contractibility to planar graphs. *Theoretical Computer Science*, 676:1–14, 2017. doi:10.1016/j.tcs.2017.02.030.

2 Takao Asano and Tomio Hirata. Edge-deletion and edge-contraction problems. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, pages 245–254, New York, NY, USA, 1982. Association for Computing Machinery. doi:10.1145/800070.802198.

3 Louis Auslander and S. V. Parter. On imbedding graphs in the sphere. *Journal of Mathematics and Mechanics*, 10(3):517–523, 1961. URL: https://www.jstor.org/stable/24900736.

4 Michael J. Bannister and David Eppstein. Crossing minimization for 1-page and 2-page drawings of graphs with bounded treewidth. *J. Graph Algorithms Appl.*, 22(4):577–606, 2018. doi:10.7155/jgaa.00479.

5 Seymour Benzer. On the topology of the genetic fine structure. *Proc. Nat. Acad. Sci.*, 45(11):1607–1620, 1959. doi:10.1073/pnas.45.11.1607.

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 Yi-Jun Chang and Hsu-Chun Yen. A new approach for contact graph representations and its applications. In *Proc. 14th Int. Symp. Algorithms and Data Structures (WADS)*, volume

9214 of *Lecture Notes in Computer Science*, pages 166–177. Springer, 2015. doi:10.1007/978-3-319-21840-3_14.

8 Steven Chaplick, Thomas C. van Dijk, Myroslav Kryven, Ji-won Park, Alexander Ravsky, and Alexander Wolff. Bundled crossings revisited. *J. Graph Algorithms Appl.*, 24(4):621–655, 2020. doi:10.7155/jgaa.00535.

9 Bruno Courcelle. The monadic second-order logic of graphs, XII: Planar graphs and planar maps. *Theoret. Comput. Sci.*, 237(1-2):1–32, 2000. doi:10.1016/S0304-3975(99)00305-9.

10 Bruno Courcelle. The monadic second-order logic of graphs, XIII: Graph drawings with edge crossings. *Theoret. Comput. Sci.*, 244(1-2):63–94, 2000. doi:10.1016/S0304-3975(00)00180-8.

11 Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach*. Cambridge University Press, 2012. doi:10.1017/CBO9780511977619.

12 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. 3.3: Planarity testing. In *Graph Drawing: Algorithms for the Visualization of Graphs*, pages 74–81. Prentice Hall, 1998.

13 Gideon Ehrlich, Shimon Even, and Robert E. Tarjan. Intersection graphs of curves in the plane. *J. Combin. Theory*, 21(1):8–20, 1976. doi:10.1016/0095-8956(76)90022-8.

14 David Eppstein, Philipp Kindermann, Stephen Kobourov, Giuseppe Liotta, Anna Lubiw, Aude Maignan, Debajyoti Mondal, Hamideh Vosoughpour, Sue Whitesides, and Stephen Wismath. On the planar split thickness of graphs. *Algorithmica*, 80(3):977–994, 2018. doi:10.1007/s00453-017-0328-y.

15 Roberto Frucht. A canonical representation of trivalent Hamiltonian graphs. *J. Graph Theory*, 1(1):45–60, 1976. doi:10.1002/jgt.3190010111.

16 Richard K. Guy and Frank Haray. On the Möbius ladders. *Canad. Math. Bull.*, 10(4):493–496, 1967. doi:10.4153/CMB-1967-046-4.

17 Petr Hlinený. Contact graphs of curves. In *Proc. Symp. on Graph Drawing (GD '95)*, volume 1027 of *Lecture Notes in Computer Science*, pages 312–323. Springer, 1995. doi:10.1007/BFB0021814.

18 Jan Kratochvíl. String Graphs. I. The number of critical nonstring graphs is infinite. *J. Combin. Theory Ser. B*, 52(1):53–66, 1991. doi:10.1016/0095-8956(91)90090-7.

19 Jan Kratochvíl. String Graphs. II. Recognizing string graphs is NP-hard. *J. Combin. Theory Ser. B*, 52(1):67–78, 1991. doi:10.1016/0095-8956(91)90091-W.

20 Jan Kratochvíl, Miroslav Goljan, and Petr Kučera. String graphs. *Rozpravy Československé Akad. Věd Řada Mat. Přírod. Věd*, 96(3):96, 1986.

21 Jǐrí Matoušek, Jaroslav Nešetřil, and Robin Thomas. On polynomial time decidability of induced-minor-closed classes. *Comment. Math. Univ. Carolin.*, 29(4):703–710, 1988. doi:10338.dmlcz/106687.

22 Miriam Münch and Ignaz Rutter. Parameterized algorithms for beyond-planar crossing numbers. In *Proc. 32nd Int. Symp. Graph Drawing and Network Visualization (GD)*, volume 320 of *LIPIcs*, pages 25:1–25:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.GD.2024.25.

23 Irina Mustata and Martin Pergel. What makes the recognition problem hard for classes related to segment and string graphs? *CoRR*, abs/2201.08498, 2022. arXiv:2201.08498.

24 Neil Robertson and Paul Seymour. Graph Minors. XIII. The disjoint paths problem. *J. Combin. Theory Ser. B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

25 Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. Recognizing string graphs in NP. *J. Computer and System Sciences*, 67(2):365–380, 2003. doi:10.1016/S0022-0000(03)00045-X.

26 F. W. Sinden. Topology of thin film RC-circuits. *Bell System Technical Journal*, 45(9):1639–1662, 1966. doi:10.1002/j.1538-7305.1966.tb01713.x.

27 Klaus Wagner. Über eine Eigenschaft der ebenen Komplexe. *Math. Ann.*, 114:570–590, 1937. doi:10.1007/BF01594196.

28 Moshe J. White. A Van Kampen type obstruction for string graphs. Electronic preprint arxiv:2308.06833, August 2023.

A Full proofs of subcubic obstacles

Proof of Proposition 17. We first show that KT_3 is not a string graph via Theorem 16. We refer to vertices of KT_3 according to their colors in Figure 7. The only choices of cubic matchings to contract are: one of the red–red edges, one of the red–blue edges, both of the red–blue edges, or one red–blue edge and the opposite red–red edge. For each of these choices, contracting the matching produces a graph that contains (as a subgraph) a subdivision of $K_{3,3}$, and so is non-planar. It follows that KT_3 is a non-string graph.

It remains to show that every proper induced minor of KT_3 is a string graph. Any vertex deletion results directly in a planar graph. Let f be an edge of KT_3; we show that there is a cubic matching M of KT_3/f such that $(KT_3/f)/M$ is planar. For convenience, in what follows we give the vertex arising from the contraction of f the color white, and the other vertices retain their colors. We have the following cases:

- If f is a red–red edge, then contracting either one of the two white–blue edges planarizes the graph.
- If f is a red–blue edge, then contracting either one of the two white–red edges planarizes the graph.
- If f is the red–yellow edge, then contracting the white–blue edge and the two red–blue edges planarizes the graph.
- If f is a blue–yellow edge, then the white vertex has three neighbors. If any of its neighbors is blue, then contracting the white–blue edge planarizes the graph. Otherwise, the white vertex has a red neighbor, and then contracting the white–red edge and the two red–blue edges planarizes the graph.

By Theorem 16, in each case KT_3/f is a string graph, from which the result follows. ◀

Proof of Proposition 18. To prove that SM_{10} is a non-string graph, by Theorem 16, it suffices to show that every cubic matching is not planarizing. Any such matching contracts some of the edges of the cycle of ten degree-3 vertices in SM_{10}, leaving it still in the form of a shorter cycle. Following the Auslander–Parter planarity testing algorithm [3, 12], for the result to be planar, the *pieces* of the graph with respect to this cycle (the two-edge paths through degree-2 vertices) must have an *interlacing graph* that is bipartite. Here, two pieces are adjacent in the interlacing graph when the vertices where they attach to cycle are in alternating order around the cycle. In SM_{10} itself, the interlacing graph is isomorphic to K_5, with each piece interlacing with two pieces that are consecutive to it around the 10-cycle and two more pieces that are non-consecutive. The non-consecutive interlacing pairs form a 5-cycle that is unaffected by contracting a matching, so the interlacing graph is non-bipartite and the contracted graph is non-planar.

To prove that SM_{10} is minimal as a non-string graph, we consider cases:

- If a degree-3 vertex is deleted from SM_{10}, the result is planar.
- If an edge incident to a degree-2 vertex is contracted, the result is a subcubic graph that can be planarized by contracting the other edge incident to the same degree-2 vertex. Therefore, the result of the first contraction is a string graph by Theorem 16.
- If a cubic edge is contracted, the resulting induced minor is not cubic, but it nevertheless has a planarizing matching depicted in Figure 8 (left). Therefore, it is a string graph.
- If a degree-2 vertex is deleted, the resulting induced minor has a planarizing matching depicted in Figure 8 (right). Therefore, it is a string graph. ◀

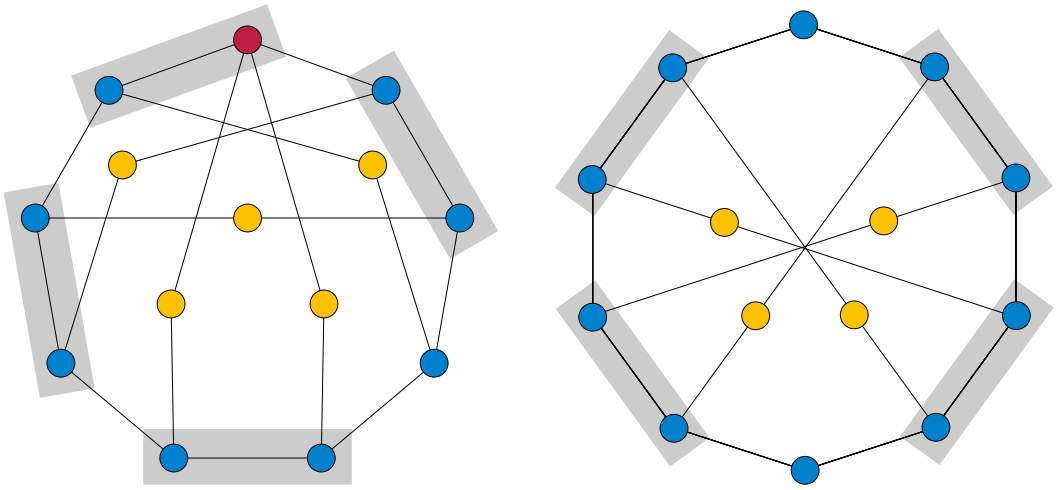


Figure 8 Left: a planarizing matching for the induced minor obtained by contracting a cubic edge of SM_{10}. Right: a planarizing matching for the induced minor obtained by deleting a degree-2 vertex of SM_{10}.

B A cubic antichain

Matoušek, Nešetřil, and Thomas [21] describe an infinite antichain in the induced minor order, consisting of planar graphs of maximum degree eight. None of these graphs can be transformed into another graph in the family by taking induced minors.

In this section we describe an infinite family of cubic graphs which form an antichain in the induced minor order. These graphs are described in LCF notation [15] by the notation $[5,-5]^n$; this means that they consist of a $2n$-vertex Hamiltonian cycle together with chords that alternate between advancing five steps forward along the cycle and five steps backward. Several well-known graphs have this form: the Heawood graph or 6-cage is $[5,-5]^7$, the Möbius–Kantor graph is $[5,-5]^8$ (Figure 9), and the smallest zero-symmetric graph is $[5,-5]^9$. We will assume $n \geq 7$ to ensure that the girth of the resulting graphs equals six.

▶ **Theorem 23.** *The graphs $[5,-5]^n$ with $n \geq 7$ form an antichain in the induced minor ordering.*

Proof. First, observe that in the neighborhood of any vertex v of one of these graphs, we can decompose the graph into three vertex-disjoint paths using edges that advance in alternating $+1$ and $+5$ steps along the Hamiltonian cycle. (Globally, these paths might connect into a single Hamiltonian cycle, as they do in Figure 9, or three disjoint cycles, depending on n.) Thus, within this neighborhood, the part of the neighborhood ahead of v around the Hamiltonian cycle and the part of the neighborhood behind v are connected by three vertex-disjoint paths. Call the edges of these paths *path edges* and the remaining edges *non-path edges*; the non-path edges are the ones between a forward and backward five-step chord of the Hamiltonian cycle, highlighted in the yellow matching in Figure 9. If any vertex of the graph were removed in the process of forming an induced minor, or if any non-path edge connecting vertices from two different local paths were contracted, the local connectivity would be reduced to two, and could not be restored by additional deletions or contractions. Therefore, no such operation can be used in forming a graph $[5,-5]^m$ as an induced minor of another graph $[5,-5]^n$.

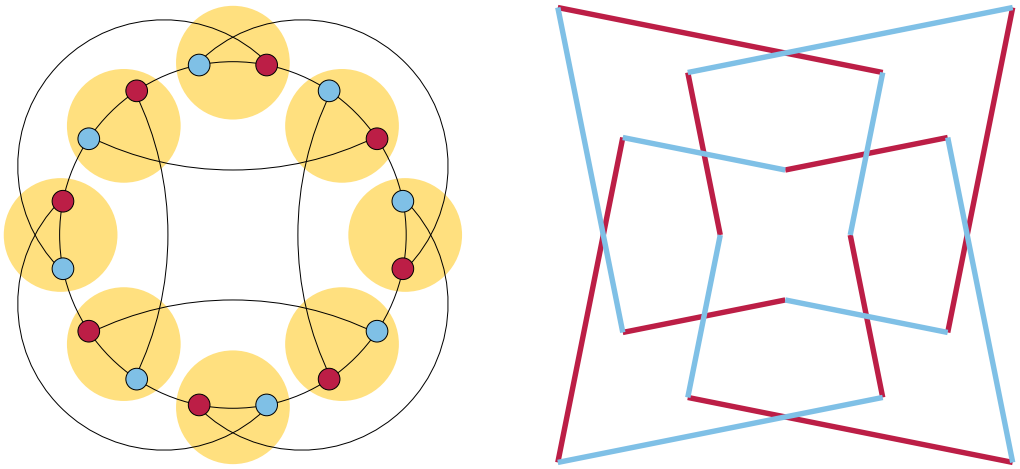


Figure 9 The Möbius–Kantor graph. Left: the vertices are arranged in order around a Hamiltonian cycle, with chords alternating five steps forward and back, according to its LCF notation $[5,-5]^8$. The non-path edges highlighted in yellow form a planarizing matching. Right: the corresponding string representation. The cycle of end-to-end contacts of strings is a different Hamiltonian cycle consisting of the path edges.

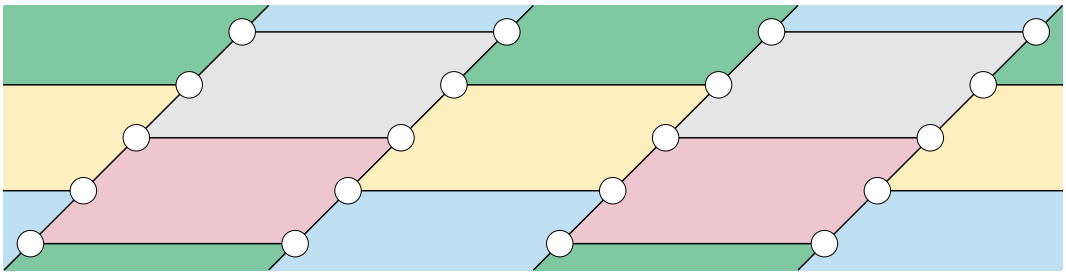


Figure 10 Torus embedding of $[5,-5]^{10}$.

It remains to show that contractions of path edges cannot produce $[5,-5]^m$ from $[5,-5]^n$. Along with the obvious 6-cycles formed by a chord and five edges of the Hamiltonian cycle, $[5,-5]^n$ has many additional 6-cycles whose edges, in terms of the difference in positions of their endpoints along the Hamiltonian cycle, follow the pattern $+1,+1,+5,-1,-1,-5$. These additional 6-cycles form the faces of a torus embedding of $[5,-5]^n$ (Figure 10), so we call them *face cycles*. If any edge of one of these face cycles is contracted, this contraction will reduce the girth of the graph, unless at least four edges of the same face are all contracted. Because every edge of $[5,-5]^n$ belongs to a face cycle, and each face cycle has exactly four path edges, the only way to avoid reducing the girth of the graph, using only contractions of uncrossed edges, is to contract all four path edges of at least one face cycle. But this also contracts an edge of the face cycle two steps ahead in the Hamiltonian cycle, which must again have all four of its path edges contracted, and so on. So any contraction of path edges that avoids reducing the girth of the graph must contract all path edges in the entire graph, which cannot result in another graph of the form $[5,-5]^m$. ◀

In an unpublished preprint, Moshe White proves that the Heawood graph $[5,-5]^7$ is not a string graph [28]. Assuming this proof to be valid, this graph provides another new cubic obstacle to string graphs:

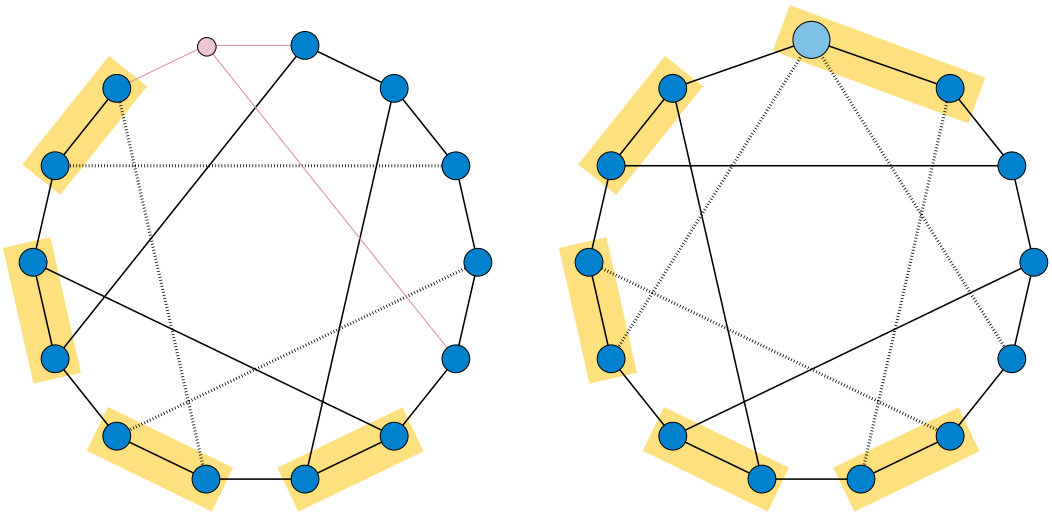


Figure 11 Planarizing matchings for proper minors of the Heawood graph. Left: Deletion of the light red vertex. Right: Contraction of an edge into the light blue degree-4 vertex.

▶ **Observation 24.** *Every proper induced minor of the Heawood graph is a string graph.*

Proof. Because the Heawood graph is both vertex-transitive and edge-transitive, we need consider only two proper induced minors, obtained by deleting one vertex or by contracting one edge. Planarizing matchings are shown for each in Figure 11, with matched pairs of vertices indicated by the yellow shading. The solid and dashed coloring of the chords of the remaining Hamiltonian cycle indicates a bipartition of the interlace graph with respect to this cycle, indicating that the result is planar. It can be drawn planarly (after contracting the matched pairs) by rerouting the dashed edges outside the Hamiltonian cycle. ◀

As Figure 9 shows, the Möbius–Kantor graph $[5,-5]^8$ is a string graph, and this can be generalized to $[5,-5]^n$ for even n:

▶ **Observation 25.** *Every graph $[5,-5]^n$ for even n is a string graph.*

Proof. It has a planarizing matching consisting of all non-path edges, such as the one in Figure 9. Contracting this matching produces a planar graph isomorphic to an $(n/2)$-antiprism. ◀

We do not know whether $[5,-5]^n$ is a string graph for other odd values of n, or if so whether a subcubic obstacle can be obtained from it.

Characterizing and Recognizing Twistedness

Oswin Aichholzer
Graz University of Technology, Austria

Alfredo García
Departamento de Métodos Estadísticos and IUMA, University of Zaragoza, Spain

Javier Tejel
Departamento de Métodos Estadísticos and IUMA, University of Zaragoza, Spain

Birgit Vogtenhuber
Graz University of Technology, Austria

Alexandra Weinberger
FernUniversität in Hagen, Germany

Abstract

In a simple drawing of a graph, any two edges intersect in at most one point (either a common endpoint or a proper crossing). A simple drawing is *generalized twisted* if it fulfills certain rather specific constraints on how the edges are drawn. An abstract rotation system of a graph assigns to each vertex a cyclic order of its incident edges. A realizable rotation system is one that admits a simple drawing such that at each vertex, the edges emanate in that cyclic order, and a generalized twisted rotation system can be realized as a generalized twisted drawing. Generalized twisted drawings have initially been introduced to obtain improved bounds on the size of plane substructures in any simple drawing of K_n. They have since gained independent interest due to their surprising properties. However, the definition of generalized twisted drawings is very geometric and drawing-specific.

In this paper, we develop characterizations of generalized twisted drawings that enable a purely combinatorial view on these drawings and lead to efficient recognition algorithms. Concretely, we show that for any $n \geq 7$, an abstract rotation system of K_n is generalized twisted if and only if all subrotation systems induced by five vertices are generalized twisted. This implies a drawing-independent and concise characterization of generalized twistedness. Besides, the result yields a simple $O(n^5)$-time algorithm to decide whether an abstract rotation system is generalized twisted and sheds new light on the structural features of simple drawings. We further develop a characterization via the rotations of a pair of vertices in a drawing, which we then use to derive an $O(n^2)$-time algorithm to decide whether a realizable rotation system is generalized twisted.

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

Keywords and phrases generalized twisted drawings, simple drawings, rotation systems, recognition, combinatorial characterization, efficient algorithms

Digital Object Identifier 10.4230/LIPIcs.GD.2025.25

Related Version *ArXiv Version*: `https://arxiv.org/abs/2508.16178v1`

Funding *Javier Tejel*: Partially supported by project E41-23R, funded by Gobierno de Aragón, and project PID2023-150725NB-I00, funded by MICIU/AEI/10.13039/501100011033.

1 Introduction

Simple drawings are drawings of graphs in which the vertices are drawn as distinct points in the plane, the edges are drawn as Jordan arcs connecting their end-vertices without passing through other vertices, and each pair of edges share at most one point (a proper crossing or a common endpoint). We consider two simple drawings the same if they are strongly isomorphic, that is, if there is a homeomorphism of the plane transforming one drawing



© Oswin Aichholzer, Alfredo García, Javier Tejel, Birgit Vogtenhuber, and Alexandra Weinberger;
licensed under Creative Commons License CC-BY 4.0



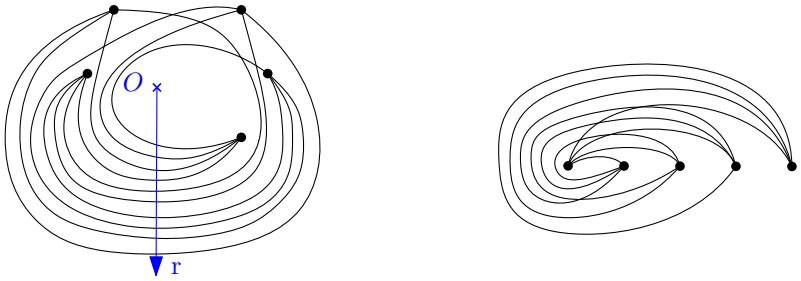


Figure 1 Two realizations of the generalized twisted rotation system of K_5. Left: Generalized twisted drawing where the origin O and ray r are depicted. Right: The classical twisted drawing.

into the other. In this work we focus on simple drawings of K_n. Unless explicitly stated otherwise, all considered drawings are of K_n and simple. A simple drawing D is *generalized twisted* if it is strongly isomorphic to a simple drawing in which there is a point O such that each ray emanating from O crosses each edge of D at most once and there exists a ray r emanating from O such that all edges of D cross r. For an example see Figure 1(left), which has been reproduced from [15].

Generalized twisted drawings have been used to find plane substructures in all simple drawings of K_n: The currently best bounds on the size of the largest plane matching and longest plane cycle and path in any simple drawing of K_n have been obtained via generalized twisted drawings [5]. Moreover, generalized twisted drawings are the largest class of drawings for which it has been shown that each drawing has exactly $2n-4$ empty triangles [14, 15], which is conjectured to be the minimum over all simple drawings [6]. There are several more interesting properties of generalized twisted drawings that have been shown already [4, 5, 15]. For example, all generalized twisted drawings contain plane cycles of length at least $n-1$ [5].

Despite the usefulness of generalized twisted drawings, their definition is very geometrical and drawing-specific. In this paper, we obtain combinatorial characterizations that no longer depend on the specific drawing, which ease the work with them and also allow for efficient recognition. Note that ad hoc it is not clear that such a combinatorial characterization needs to exist. The study of special classes of simple drawings [3, 11, 12, 13] and their combinatorial characterizations [8, 9, 10, 22, 23] has a strong tradition in the research of simple drawings and contributes significantly to the understanding simple drawings in general.

A widely used combinatorial abstraction of drawings are rotation systems. The *rotation of a vertex* in a drawing is the cyclic order of its incident edges. The *rotation system of a simple drawing* is the collection of the rotations of all vertices. Rotation systems are especially useful for describing simple drawings of complete graphs, since two simple drawings of K_n have the same crossing edge pairs if and only if they have the same or inverse rotation system [7, 16, 17, 19]. Rotation systems of graphs can also be considered without a concrete drawing. An *abstract rotation system* of a graph G gives, for every vertex of G, a cyclic order of its adjacent vertices. An abstract rotation system is called *realizable* if there exists a simple drawing with that rotation system. For abstract rotation systems of complete graphs, realizability can be checked in polynomial time. Specifically, an abstract rotation system of K_n is realizable if and only if every subrotation system induced by five vertices is realizable. This follows from a combinatorial characterization via subrotation systems of size six [20, Theorem 1.1.] together with a complete enumeration of all realizable rotation systems of K_n for $n \leq 9$ [2].

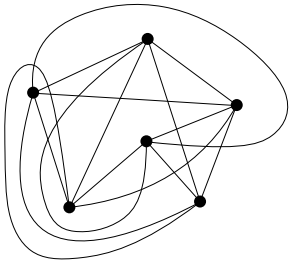


Figure 2 A simple drawing of K_6 whose rotation system is not generalized twisted, even though all subrotation systems on five vertices are.

In this work, we use rotation systems to characterize generalized twisted drawings. An abstract rotation system of a graph is *generalized twisted* if there is a generalized twisted drawing realizing this rotation system. As our first main result, we show that generalized twisted rotation systems of K_n can also be characterized via subrotation systems of size five. The main benefit of this result is to have a drawing-independent (and concise) characterization of generalized twistedness. The substructures to be considered are of constant size and straightforward to be checked, which improves the insight into generalized twisted drawings.

▶ **Theorem 1.** *Let R be an abstract rotation system of K_n with $n \geq 7$. Then R is generalized twisted if and only if every subrotation system induced by five vertices is generalized twisted.*

This characterization of generalized twisted rotation systems nicely complements the characterization of all realizable simple drawings in general, which can also be done by considering subrotation systems of size five [2, 20]. Per se, such characterizations of bounded complexity do not always need to exist. For example, if the combinatorics of an (abstract) set of points is given by their triple orientations – so-called abstract order types – no simple characterization to decide their realizability as a point set in the plane can exist, as this decision problem is known to be $\exists\mathbb{R}$-complete; see the recent survey [24] for details.

The main part of Theorem 1 is the "if" part. The "only-if" part follows from the fact that subdrawings of generalized twisted drawings are generalized twisted by definition. As mentioned in the article introducing generalized twisted drawings [4], there is exactly one generalized twisted rotation system T_5 of K_5, namely, the one of the *twisted* drawing [18, 21] of K_5, which gave rise to the name "generalized twisted"; see Figure 1(right) for an illustration as in [21]. We remark that Theorem 1 cannot be extended to $n \leq 6$. Figure 2 depicts a drawing from [4], where it has been shown that the rotation system of this drawing is not generalized twisted, although all its subrotation systems are (generalized) twisted.

Our second main result is a characterization of generalized twisted drawings via rotation systems and crossings as stated in Theorem 2 below and depicted on an example in Figure 3.

▶ **Theorem 2.** *Let R be the rotation system of a simple drawing D of K_n and let V be the set of vertices of D. Then R is generalized twisted if and only if there exist two vertices v_1 and v_2 in V and a bipartition $A \cup B$ of the vertices in $V \setminus \{v_1, v_2\}$, where some of A or B can be empty, such that:*

- **(i)** *For every pair of vertices a and a' in A, the edge aa' crosses v_1v_2.*
- **(ii)** *For every pair of vertices b and b' in B, the edge bb' crosses v_1v_2.*
- **(iii)** *For every vertex $a \in A$ and every vertex $b \in B$, the edge ab does not cross v_1v_2.*
- **(iv)** *Beginning at v_2, in the rotation of v_1, all vertices in B appear before all vertices in A.*
- **(v)** *Beginning at v_1, in the rotation of v_2, all vertices in B appear before all vertices in A.*

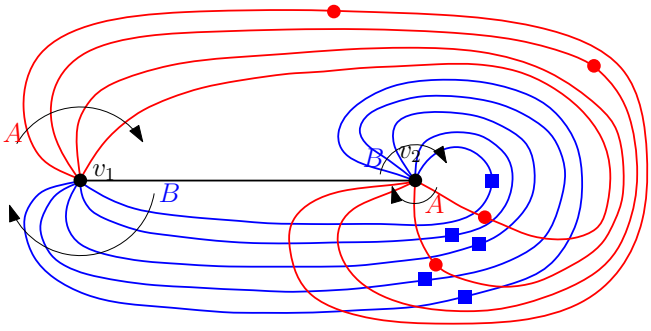


Figure 3 Illustration of Theorem 2: Possible rotations of v_1 and v_2. The vertices of partition set A are drawn as red disks and the vertices of partition set B as blue squares.

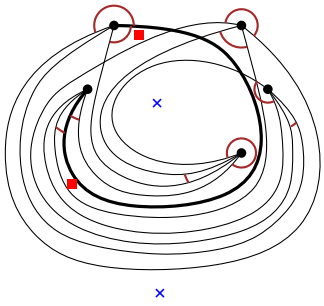


Figure 4 A realization of the generalized twisted rotation system T_5 of K_5. The pairs of antipodal vi-cells are marked with red squares and blue crosses, the maximum crossing edge is bold, and empty triangles are indicated by (brown) circular arcs.

The main implication of Theorem 2 is that it leads to fast algorithms for recognizing generalized twisted rotation systems. If it is already known that a rotation system is realizable (which needs $O(n^5)$ time to be checked [2, 20]), we can even use it to obtain a quadratic time recognition algorithm.

▶ **Theorem 3.** *Let R be an abstract rotation system of K_n. Then it can be decided in $O(n^5)$ time whether R is generalized twisted. If R is known to be realizable, then deciding whether it is also generalized twisted can be done in $O(n^2)$ time.*

Outline. We start by introducing some notation and showing first properties in Section 2. Sections 3 and 4 are devoted to proving the two characterizations, Theorem 1 and Theorem 2, respectively. In Section 5, we prove Theorem 3 by providing the algorithms. We conclude in Section 6 with some open questions.

2 Preliminaries

Given a simple drawing D, the *star at v*, denoted by $S(v)$, is the set of edges incident to v in D. For a set V of vertices of D, we denote by $D \setminus V$ the drawing D without the vertices in V and without all edges incident to a vertex in V. A *triangle* Δ in D is the simple closed curve formed by three vertices and the three edges connecting them. Every triangle partitions the plane and hence D into two regions which we call the *sides* of Δ. The whole drawing D partitions the plane into regions called *cells*.

In our proofs, we heavily rely on antipodal vi-cells, empty star triangles, and maximum crossing edges, all of which we describe in the following paragraphs.

Antipodal vi-cells. A cell that is incident to a vertex (that is, has the vertex on its boundary) is called vertex-incident cell or *vi-cell*. We observe that if there is a vi-cell at v containing the initial parts of vu and vw in one drawing of a rotation system, then vu and vw are neighboring in the rotation of v. Since the latter property depends only on the rotation system of the drawing, there is a vi-cell at v in every drawing of this rotation system with initial parts of vu and vw on the boundary. Note that the remainder of the boundary of the vi-cell is not necessarily identical across different drawings of the rotation system. Two cells are *antipodal* if for every triangle, the two cells are on different sides. A vi-cell that is antipodal to some other vi-cell is called a *via-cell*. (In Figure 4, the two pairs of via-cells are marked.) Since the set of crossing edge pairs determines the rotation system and vice versa [17, 19], also the existence of via-cells is a property of the rotation system. A vertex incident to a via-cell is called a *via-vertex*.

We will need the following known theorem [4, Theorem 15][1].

▶ **Theorem 4** ([4]). *The rotation system of a simple drawing D of K_n is generalized twisted if and only if D contains a pair of antipodal vi-cells.*

Note that the characterization in Theorem 4 implies that a rotation system is generalized twisted if and only if *all its realizations as simple drawings* contain a pair of antipodal vi-cells.

Empty star triangles. A triangle Δ with vertices x, y and z is a *star triangle* at x if the star $S(x)$ does not cross the edge yz; it is *empty* if one side of Δ contains no vertices of $D \setminus \Delta$ (and hence the other side contains all of them). (In Figure 4, each circular arc between two edges at a vertex indicates that the triangle containing those two edges is an empty star triangle at that vertex.) We call two star triangles at x *adjacent* if they are both incident to some edge xy and span disjoint angles at x. We will use the following results on empty star triangles in simple and generalized twisted drawings.

▶ **Lemma 5** ([6]). *Let v be a vertex of a simple drawing D of K_n. Then D contains at least two empty star triangles at v. If vxy is a star triangle at v then it is empty if and only if the vertices x and y are consecutive in the rotation of v.*

▶ **Lemma 6** ([15]). *Let D be a generalized twisted drawing of K_n and let v be a vertex of D. Then there are exactly two empty star triangles at v. If (C_1, C_2) is a pair of antipodal vi-cells in D, then one of the two empty star triangles at v contains C_1 on the empty side and the other triangle contains C_2 on the empty side.*

Maximum crossing edges. An edge v_1v_2 is *maximum crossing* if v_1v_2 crosses every edge not in $S(v_1) \cup S(v_2)$. (For example, the bold edge in Figure 4 is maximum crossing.) Any simple drawing has at most one maximum crossing edge, as the following lemma states.

▶ **Lemma 7.** *Any simple drawing of K_n with $n \geq 5$ has at most one maximum crossing edge.*

[1] The original formulation uses drawings and weak isomorphism. Our formulation is equivalent due to the equivalence of rotation systems and weak isomorphism [17, 19].

Lemma 7 holds for $n = 5$ because none of the five simple drawings of K_5 (up to strong isomorphism) [6] contains more than one maximum crossing edge. Using this fact, the lemma can be proven for $n > 5$ by contradiction, assuming that a simple drawing contains several maximum crossing edges.

We next observe in Lemma 8 that maximum crossing edges imply generalized twisted rotation systems. Figure 5 illustrates the lemma.

▶ **Lemma 8.** *Let D be a simple drawing of K_n that has a maximum crossing edge, v_1v_2. Let C_1 and C_1' be the two cells incident to v_1 along the edge v_1v_2 and let C_2 and C_2' be the two cells incident to v_2 along the edge v_1v_2 such that when going along the edge from v_1 to v_2, the cell C_1 is on the other side of v_1v_2 than the cell C_2 (and consequently C_1' is on the other side of v_1v_2 than C_2'). Then D is generalized twisted and (C_1, C_2) and (C_1', C_2') are two pairs of antipodal vi-cells.*

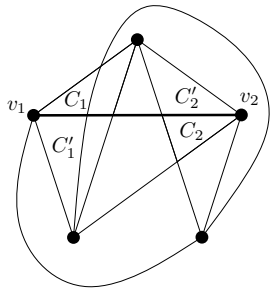


Figure 5 A maximum crossing edge v_1v_2 (in bold) and the two pairs, (C_1, C_2) and (C_1', C_2'), of antipodal vi-cells.

Lemma 8 can be shown by taking an arbitrary triangle of the generalized twisted drawing and considering where the cells adjacent to the edge v_1v_2 and its endpoints must lie with respect to this triangle in order to allow the edge v_1v_2 to cross all non-adjacent edges.

3 Characterization via subrotation systems induced by five vertices

In this section, we prove Theorem 1 by induction. To this end, we use the following theorem, whose proof we sketch in Section 3.1.

▶ **Theorem 9.** *Let R be an abstract rotation system of K_n with $n \geq 12$ such that every induced subrotation system on at most $n - 1$ vertices is generalized twisted. Then R is generalized twisted.*

Proof of Theorem 1. By definition, if R is generalized twisted, then any subrotation system induced by five vertices is generalized twisted. We prove the converse: if every subrotation system induced by five vertices is generalized twisted, then R is generalized twisted.

For $7 \leq n \leq 11$, the theorem is verified by computer. Following the lines of [1, 5], we compute all generalized twisted rotation systems of K_n for n up to 15 in the following way. First, we use the method of [2] to generate all realizable rotation systems, but discard those with subrotation systems induced by five vertices different to T_5, thus computing all candidate rotation systems for generalized twisted drawings. In each candidate rotation system, we then search for a pair of antipodal vi-cells. By Theorem 4, this identifies all generalized twisted rotation systems. Any rotation system contradicting Theorem 1 does not have antipodal vi-cells. The computational results give only one rotation system of size $n = 6$ (drawn in Figure 2) that is not generalized twisted.

For $n \geq 12$, we use induction. Every induced subrotation system of R on at most $n-1$ vertices is generalized twisted by induction. Thus, R is generalized twisted by Theorem 9. ◀

3.1 Proof sketch of Theorem 9

Let R be as in Theorem 9 and let D be a realization of R, which exists since any abstract rotation system of K_n is realizable if all subrotation systems induced by five vertices are [2, 20]. Since each induced and proper subdrawing of D has a generalized twisted rotation system, it also has a pair of antipodal vi-cells by Theorem 4. We use this to show that also D has such a pair and thus, by Theorem 4, R is generalized twisted. The idea is to locate antipodal vi-cells of D using those in its induced subdrawings and empty triangles. The following lemma can be shown using Lemma 5 (for a lower bound) and Lemma 6 (for an upper bound).

▶ **Lemma 10.** *Let D be a drawing of K_n with $n \geq 5$ such that every induced and proper subdrawing has a generalized twisted rotation system. Then at any vertex of D, there are exactly two empty star triangles.*

We separately consider the case where D contains a vertex u with two adjacent empty star triangles at u and the case that there is no such vertex.

Case 1: Two adjacent empty star triangles at a vertex u.
This is the easy case. We first show the following lemma on generalized twisted rotation systems of K_n. Its proof is based on considering two adjacent empty star triangles at a vertex v_1 and analyzing all possibilities of how to extend these triangles to a drawing of T_5 and then to a drawing of K_6 with a generalized twisted rotation system.

▶ **Lemma 11.** *Let v_1 and v_2 be two vertices of a simple drawing of K_n with generalized twisted rotation system such that the two empty star triangles at vertex v_1 are sharing the edge v_1v_2. Then the following properties hold.*

1. *The edge v_1v_2 is maximum crossing.*
2. *The two empty star triangles at v_1 are also empty star triangles at v_2.*
3. *For each of the two empty star triangles at v_1, the vertices v_1 and v_2 are adjacent in the rotation of the third vertex of the triangle in such a way that all edges emanate from the third vertex in the empty side of the triangle.*

Then we consider the drawing D and the two adjacent empty star triangles at u in D. As every proper subdrawing of D has a generalized twisted rotation system, Lemma 11 implies that the edge shared by the two adjacent empty star triangles at u is maximum crossing. Thus, the rotation system of D is generalized twisted by Lemma 8.

Case 2: No vertex with two adjacent empty star triangles.
This is the involved case. Let v_1 be an arbitrary vertex of D and let $\Delta_1 = v_1u_1u_1'$ and $\Delta_2 = v_1u_2u_2'$ be the (nonadjacent) empty star triangles at v_1. Our goal is to show that using via-cells of some induced and proper subdrawings of D, we can find a pair (C_1, C_2) of vi-cells in D such that C_1 lies in Δ_1 and C_2 in Δ_2, and prove that C_1 and C_2 are antipodal.

Let V_0 be the set of vertices of Δ_1 and Δ_2, and let D_0 be the subdrawing induced by V_0. Since the rotation system of D_0 is generalized twisted and has only five vertices, we have all the information about that drawing. This includes in particular the maximum crossing edge of D_0, which we denote by m, the two pairs of antipodal vi-cells of D_0, and the empty star triangles. As Δ_1 and Δ_2 are not adjacent, v_1 is one of the vertices $\{a, b, d\}$ in Figure 6.

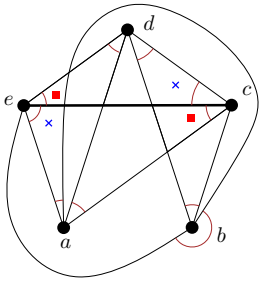


Figure 6 A drawing of the unique generalized twisted rotation system T_5 of K_5, which is strongly isomorphic to the one in Figure 1, but redrawn to make empty star triangles and the relevant cells more visible. Empty star triangles at a vertex are marked with brown arcs at the angle that spans the triangle. The two pairs of antipodal vi-cells are marked with red squares and blue crosses.

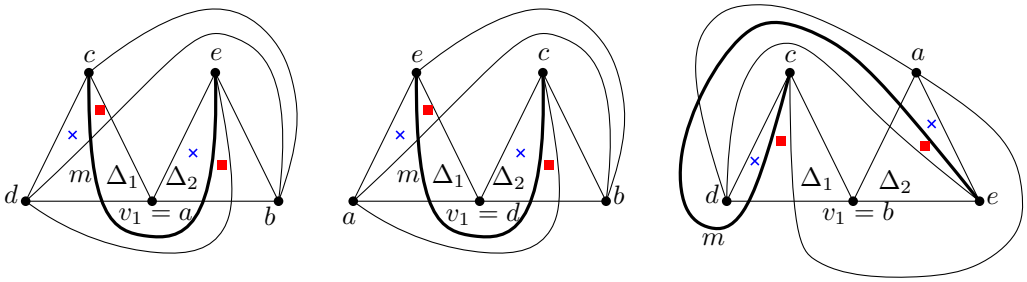


Figure 7 Three drawings realizing T_5 in such a way that v_1 is drawn in the center. The underlying unlabeled drawings are strongly isomorphic. The labels correspond to those in Figure 6 with $v_1 = a$ (left), $v_1 = d$ (middle), or $v_1 = b$ (right). The maximum crossing edge m is drawn bold, the two pairs of antipodal vi-cells are marked with red squares and blue crosses.

Observe that the vertices a and d behave analogously to each other while vertex b behaves differently (see Figure 7). We argue via the triangles Δ_1 and Δ_2 separately but simultaneously, because we will consider only one triangle but require that the other triangle has the same properties. We sketch the case where v_1 is a or d and focus mostly on Δ_1. The properties for Δ_2 can be shown analogously and the case of v_1 behaving like b then follows immediately.

For identifying all pairs of antipodal vi-cells in subdrawings of D, we state in Lemma 12 properties of simple drawings with more than one such pair.

▶ **Lemma 12.** *Let D be a simple drawing of K_n with different antipodal vi-cell pairs (C_1, C_2) and (C_1', C_2'). Then*

1. *For one of the vertices on the boundary of a cell in $\{C_1, C_2, C_1', C_2'\}$, there are two empty star triangles at that vertex that share an edge.*
2. *There is a maximum crossing edge which is the shared edge of the two empty star triangles at one of the endpoints of the edge.*
3. *The via-cells are adjacent to the maximum crossing edge (as described in Lemma 8).*
4. *D has only those two (and no more) pairs of antipodal vi-cells.*

Proof Sketch. The proof of (1) is by contradiction. Assuming that the two empty star triangles at any vertex on the boundary of a cell in $\{C_1, C_2, C_1', C_2'\}$ are not adjacent, we can find an induced subdrawing on 5 or 6 vertices that is not generalized twisted, a contradiction. (2) and (3) follow from Lemma 11, as by (1) there are two empty star triangles that are adjacent. (4) follows from the fact that if there is a third pair of antipodal vi-cells, then D would contain two maximum crossing edges, contradicting Lemma 7. ◀

Let V' be a nonempty subset of $V \setminus V_0$. By Lemma 12 and Lemma 6, one of the following holds for the subdrawing $D \setminus V'$ of D:

1. $D \setminus V'$ contains exactly one pair of antipodal vi-cells. One of those cells is in the empty side of triangle Δ_1 and the other one is in the empty side of triangle Δ_2.
2. $D \setminus V'$ contains exactly two pairs of antipodal vi-cells. For each pair, one cell is in the empty side of Δ_1 and the other one is in the empty side of Δ_2. Further, for each of Δ_1 and Δ_2, the two vi-cells in its empty side are adjacent along the maximum crossed edge of $D \setminus V'$ and incident to the endpoints of this edge.

The next lemma states the possibilities of how *arbitrary antipodal vi-cell pairs* of two subdrawings $D\setminus\{v\}$ and $D\setminus\{w\}$ of D can be contained in *all pairs* of the subdrawing $D\setminus\{v,w\}$. The proof uses observations on the containment of cells of a drawing D' in cells of its subdrawings and consequences of two cells being antipodal. We denote by $(C_1^{v_1v_2\cdots}, C_2^{v_1v_2\cdots})$ an antipodal vi-cell pair in the drawing $D\setminus\{v_1, v_2, \ldots\}$.

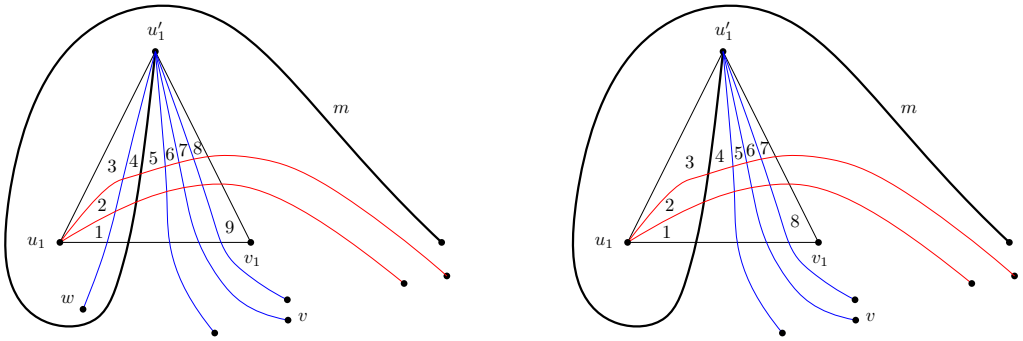


Figure 8 The grid cells of D and $D\setminus\{w\}$ inside triangle Δ_1.

▶ **Lemma 13.** *For any pair (C_1^v, C_2^v) of antipodal vi-cells in $D\setminus\{v\}$ and any pair (C_1^w, C_2^w) of antipodal vi-cells in $D\setminus\{w\}$, then*

1. *If $D\setminus\{v,w\}$ has **only one antipodal vi-cell pair**, (C_1^{vw}, C_2^{vw}), then (C_1^v, C_2^v) and (C_1^w, C_2^w) both are contained in (C_1^{vw}, C_2^{vw}).*
2. *If $D\setminus\{v,w\}$ has **two antipodal vi-cell pairs** (C_1^{vw}, C_2^{vw}) and $(C_1'^{vw}, C_2'^{vw})$, then either (C_1^v, C_2^v) and (C_1^w, C_2^w) both are contained in one of (C_1^{vw}, C_2^{vw}) and $(C_1'^{vw}, C_2'^{vw})$, or (C_1^v, C_2^v) is contained in one of (C_1^{vw}, C_2^{vw}) and $(C_1'^{vw}, C_2'^{vw})$, and (C_1^w, C_2^w) in the other.*

We identify grid structures in the empty sides of Δ_1 and Δ_2 in D that we will work with to locate candidates for antipodal vi-cells of D. As Δ_1 is an empty star triangle at v_1 and as D is a simple drawing, no edge incident to v_1 can cross an edge of Δ_1. However, every generalized twisted drawing of K_4 contains a crossing [5]. Hence, for every vertex $v \notin \Delta_1$, exactly one of vu_1 or vu_1' crosses Δ_1 (on v_1u_1' or v_1u_1, respectively). We denote this edge by r_v. The set of edges r_v for every v form a grid in the empty side of Δ_1 in D. We denote the cells of this grid that have some vertex (of Δ_1) on the boundary as *grid cells*; see Figure 8. Grid cells in Δ_2 are defined analogously. Note that grid cells may consist of several cells of D, as edges of D can cross through Δ_1 and/or Δ_2.

Next, we obtain an order on the grid cells of Δ_1 (and analogously on Δ_2). Consider the dual graph of the grid cells (where the vertices are the grid cells and two vertices share an edge if the corresponding grid cells are adjacent). This dual graph is either a path or a path plus an isolated vertex (e.g. grid cell 9 in Figure 8 left). We label the grid cells $G_1, \ldots, G_l$, along this dual path, beginning with the grid cell with vertex u_1 and adjacent to edge u_1v_1,

and finishing with the cell that has v_1 on its boundary; see Figure 8 left for an example. The *distance* between two grid cells G_i, G_j is the number of edges r_v separating G_i and G_j. If edge r_v is incident to both G_i and G_{i+1}, we say that G_i and G_{i+1} are *glued along* r_v. We denote the two grid cells that are glued along the edge m (which is the maximum crossed edge of D_0) by G_c and G_{c+1} (cells $4, 5$ in Figure 8 left). We say that grid cell G_i is *placed before* m if $i \leq c$; and *placed after* m otherwise.

When removing a vertex $v \in V \setminus V_0$, we obtain a grid in $D\backslash\{v\}$ whose only difference to the grid in D is that two grid cells of D that are adjacent at the edge r_v become one grid cell in $D\backslash\{v\}$, and that some grid cells may enlarge their size, depending on the removed vertex; see Figure 8 right for an illustration. For each via-cell C^v of $D\backslash\{v\}$ in the empty side of Δ_1, we denote by G^v the grid cell of $D\backslash\{v\}$ containing C^v. Observe that G^v is either an (enlarged) grid cell of D or two such grid cells glued along an edge r_v. With the following lemma, we obtain that the grid cells containing via-cells of the subdrawings have common intersections in the grid cells of D.

▶ **Lemma 14.** *Let $w_1, w_2, \ldots, w_s$ be the set of vertices in $V\setminus V_0$ such that for every w_i, the grid cell G^{w_i} in $D\backslash\{w_i\}$ is placed before m. Let $x_1, x_2, \ldots, x_t$ be the set of vertices in $V\setminus V_0$ such that for every x_i, the grid cell G^{x_i} in $D\backslash\{x_i\}$ is placed after m. Then*

1. *If $s \geq 2$, the common intersection of $G^{w_1}, G^{w_2}, \ldots, G^{w_s}$ is a grid cell of D placed before m.*
2. *If $t \geq 2$, the common intersection of $G^{x_1}, G^{x_2}, \ldots, G^{x_t}$ is a grid cell of D placed after m.*
3. *It $s \geq 2$ and $t \geq 2$, then the two obtained grid cells of D are G_c and G_{c+1}, respectively.*

Proof Sketch. (1) Given w_i and w_j, we show that $G^{w_i} \cap G^{w_j}$ is a grid cell of D by analyzing the intersections among G^{w_i}, G^{w_j} and $G^{w_i w_j}$, for the different possibilities of the positions of r_{w_i} and r_{w_j}. Then we show by contradiction that $G^{w_1}, G^{w_2}, \ldots, G^{w_s}$ have a common intersection, which is a grid cell of D placed before m. Assuming that $G^{w_i} \cap G^{w_j} \neq G^{w_j} \cap G^{w_k}$, we show that G^{w_i} and G^{w_k} cannot be contained in $G^{w_i w_k}$, a contradiction. (2) is proved in a similar way. To prove (3), we show by contradiction that G_c (respectively G_{c+1}) is contained in G^{w_i} (respectively G^{x_i}) for any i, so G_c and G_{c+1} must be the common grid cells. ◀

Next we show that with enough vertices in total, we also obtain vi-cells of D in the (shared) grid cells we found in D.

▶ **Lemma 15.** *Suppose that for the via-cells $C^{w_1}, C^{w_2}, \ldots, C^{w_s}$ with $s \geq 2$, the corresponding grid cells $G^{w_1}, G^{w_2}, \ldots, G^{w_s}$ in $D\backslash\{w_1\}, D\backslash\{w_2\}, \ldots, D\backslash\{w_s\}$, respectively, contain a common grid cell G of D. Then $C^{w_1}, C^{w_2}, \ldots, C^{w_s}$ also contain a common vi-cell C of D.*

Proof Sketch. We first prove that if G^{w_i} and G^{w_j} contain a common grid cell G, then the interior of C^{w_i} (or C^{w_j}) and the interior of G are not disjoint. Using this result, if G has only one vertex on its boundary, then the lemma trivially holds. If G contains two vertices on its boundary, we partition $C^{w_1}, C^{w_2}, \ldots, C^{w_s}$ into three sets S_1, S_2, and S_3, where S_1 contains the cells C^{w_i} that are incident to only one of the vertices, S_2 contains the cells C^{w_i} that are incident to only the other vertex, and S_3 contains the cells C^{w_i} that are incident to both vertices. We then prove that one of S_1 and S_2 must be empty, so $C^{w_1}, C^{w_2}, \ldots, C^{w_s}$ must all be incident to the same vertex of G and share a common vi-cell C of D. ◀

The following lemma shows that D contains a pair of antipodal vi-cells.

▶ **Lemma 16.** *For $n \geq 12$, there exists a pair (C, C') of antipodal vi-cells in D.*

Proof Sketch. For $i = 1, \ldots, 7$, let w_i be a vertex not in V_0 and let $(C_1^{w_i}, C_2^{w_i})$ be a pair of antipodal vi-cells in the subdrawing $D\backslash\{w_i\}$. For any pair of antipodal vi-cells of V_0, one of the cells is in Δ_1 and the other in Δ_2. Hence, by Lemma 13, $C_1^{w_i}$ must lie in Δ_1 and $C_2^{w_i}$ in

Δ_2 for any i. Using Lemmas 14 and 15, we show that at least four of the vi-cells $C_1^{w_i}$ share a common vi-cell C in Δ_1 and a common vi-cell C' in Δ_2 for the corresponding vi-cells $C_2^{w_i}$. By the antipodality of the pairs $(C_1^{w_i}, C_2^{w_i})$ that contain (C, C'), we prove the antipodality of (C, C') by showing for all triangles of D that C and C' are on different sides. ◀

4 Characterization via bipartition

This section is devoted to the proof of Theorem 2. Before starting with the proof, we introduce the concept of compatibility of vertices. Given a realizable rotation system of K_n, let $R_{v_1} = \{v_2 = w_1, w_2, \ldots, w_{n-1}\}$ and $R_{v_2} = \{v_1 = t_1, t_2, \ldots, t_{n-1}\}$ be the (clockwise) rotations of v_1 and v_2, respectively. We say that v_1 and v_2 are *compatible* (or that v_1 is compatible with v_2) if $\{v_1, t_2, \ldots, t_{n-1}\} = \{v_1, w_i, w_{i-1}, \ldots, w_2, w_{n-1}, w_{n-2}, \ldots, w_{i+1}\}$, for some $i \in \{1, 2, \ldots, n\}$. Note that if v_1 and v_2 are compatible and i is 1 or n, then $\{v_1, t_2, \ldots, t_{n-1}\} = \{v_1, w_{n-1}, w_{n-2}, \ldots, w_2\}$.

Next we consider pairs of antipodal vi-cells in simple drawings (and thus in generalized twisted drawings) and show that each such pair contains a pair of compatible vertices. For any antipodal vi-cell pair (C_1, C_2) in a simple drawing of K_n, there exist two vertices $v_1 \neq v_2$ such that v_1 and v_2 are incident to C_1 and C_2, respectively [4]. The following lemma implies that v_1 and v_2 are compatible.

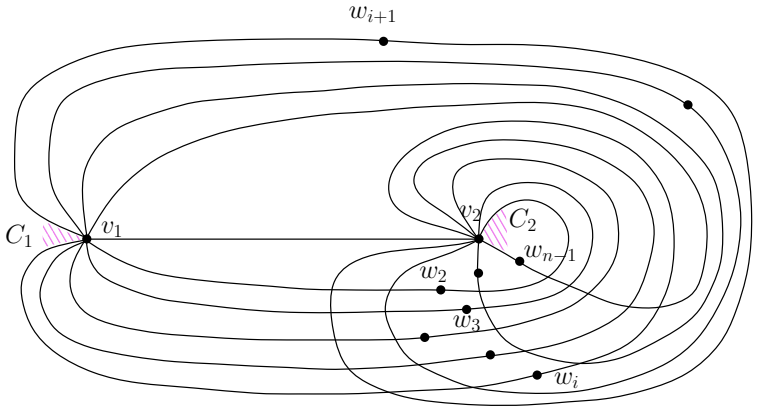


Figure 9 The rotations of v_1 and v_2.

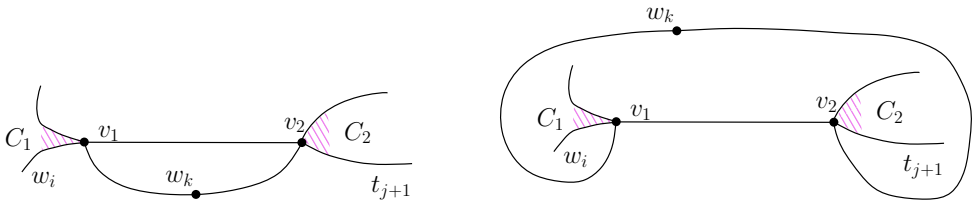


Figure 10 w_k cannot appear after C_2 around v_2.

▶ **Lemma 17.** *Let D be a realization of a generalized twisted rotation system of K_n with $n \geq 3$. Let (C_1, C_2) be a pair of antipodal vi-cells of D, and let v_1 and v_2 be vertices of D incident to C_1 and C_2, respectively, with $v_2 \neq v_1$. Assume that $R_{v_1} = \{v_2 = w_1, w_2, \ldots, w_{n-1}\}$ is the (clockwise) rotation of v_1, and that the edges v_1w_i and v_1w_{i+1} define part of the boundary of C_1, for some $i \in \{1, \ldots, n-1\}$. Then the following statements hold.*

(i) *If $i \in \{2, 3, \ldots, n-2\}$ then the (clockwise) rotation of v_2 is $\{v_1, w_i, w_{i-1}, \ldots, w_2, w_{n-1}, w_{n-2}, \ldots, w_{i+1}\}$ and the edges v_2w_2 and v_2w_{n-1} define part of the boundary of C_2.*

(ii) *If $i = 1$ then the (clockwise) rotation of v_2 is $\{v_1, w_{n-1}, w_{n-2}, \ldots, w_2\}$ and the edges v_2v_1 and v_2w_{n-1} define part of the boundary of C_2.*

(iii) *If $i = n-1$ then the (clockwise) rotation of v_2 is $\{v_1, w_{n-1}, w_{n-2}, \ldots, w_2\}$ and the edges v_2v_1 and v_1w_2 define part of the boundary of C_2.*

Proof Sketch. We sketch the proof of (i); the proofs of (ii) and (iii) are very similar. Figure 9 shows an example of the rotations of v_1 and v_2. Assume that the clockwise rotation of v_2 is $\{v_1 = t_1, t_2, \ldots, t_{n-1}\}$ and that the edges v_2t_j and v_2t_{j+1}, for some $j \in \{1, \ldots, n-1\}$, define part of the boundary of C_2. To place w_k in the rotation of v_1 and v_2 with respect to C_1 and C_2, we consider for w_k with $2 \leq k \leq i$ the triangle spanned by v_1, v_2 and w_k. As C_1 and C_2 are antipodal, they must lie on different sides of the triangle. This is not possible if w_k corresponds to a vertex $t_{k'}$ with $j+1 \leq k' \leq n-1$ (see Figure 10 for a depiction). Thus, if a vertex appears before (via analogous arguments after) C_1 around v_1 clockwise from v_2, then it also appears before (after) C_2 around v_2 clockwise from v_1.

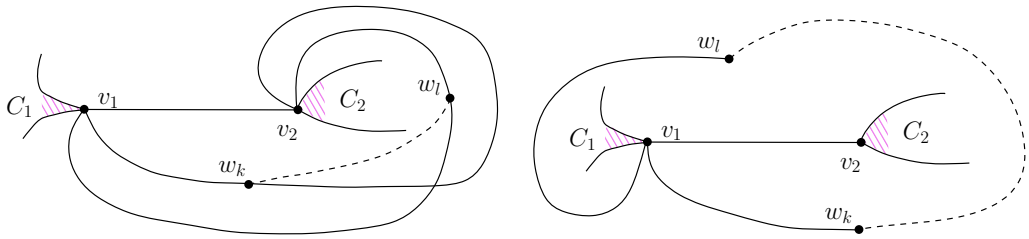


Figure 11 w_k cannot appear before w_l around v_2.

For two vertices w_k and w_l with $2 \leq k < l \leq i$, we show that w_l appears before w_k in the clockwise rotation of v_2 from v_1 by contradiction. Consider the edges v_1w_l and v_2w_l. If w_k appears before w_l in the clockwise rotation of v_2 from v_1, then the edges v_1w_l and v_2w_l emanate in different sides of the triangle $v_1v_2w_k$ from v_1 and v_2. From simplicity of the drawing and the fact that each realization of a generalized twisted rotation system of K_4 contains exactly one crossing [5], we can determine how that edge w_lw_k must behave (see Figure 11 for a depiction). We then show that with this drawing, C_1 and C_2 must lie on the same side of the triangle $v_1w_kw_l$, which contradicts that they are antipodal cells. ◀

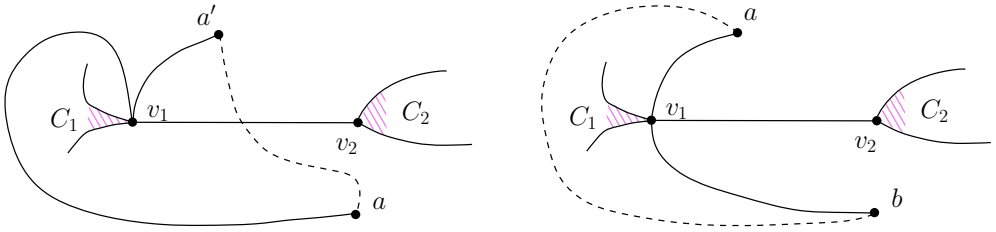


Figure 12 Illustrating the proof of Theorem 2 (i) and (iii).

Using Lemma 17 we can prove Theorem 2.

Proof Sketch of Theorem 2. If R is generalized twisted, then there exists a pair (C_1, C_2) of antipodal vi-cells in D, and we show that a bipartition of the vertices satisfies (i)–(v). Let v_1 be a vertex of V incident to C_1 and let $v_2 \neq v_1$ be a vertex of V incident to C_2.

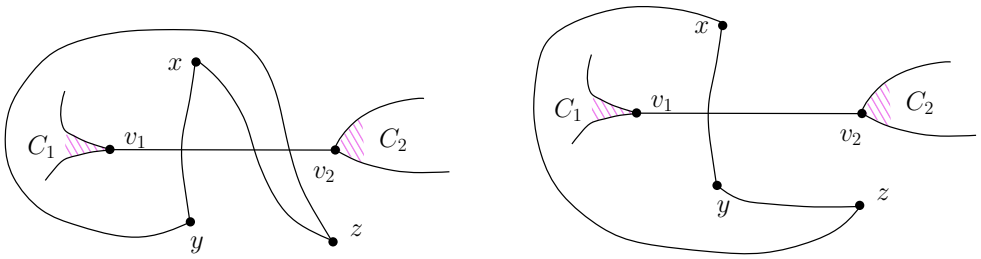


Figure 13 C_1 and C_2 are on different sides of Δxyz. Left: x, y, and z belong to the same set of the bipartition. Right: x and y belong to one of the sets and z to the other.

Suppose that $\{v_2 = w_1, w_2, \ldots, w_{n-1}\}$ is the (clockwise) rotation of v_1 such that v_1w_i and v_1w_{i+1} define part of the boundary of C_1, for some $i \in \{1, \ldots, n-1\}$. Let A and B be defined as follows: if $i \notin \{1, n-1\}$, set $A := w_{i+1}, ..., w_{n-1}$, $B := w_2, ..., w_i$; if $i = 1$, set $A := \{v_1, w_{n-1}, w_{n-2}, \ldots, w_2\}$, $B := \emptyset$; if $i = n-1$, set $A := \emptyset$, $B := \{v_1, w_{n-1}, w_{n-2}, \ldots, w_2\}$. We show that this bipartition satisfies (i)–(v).

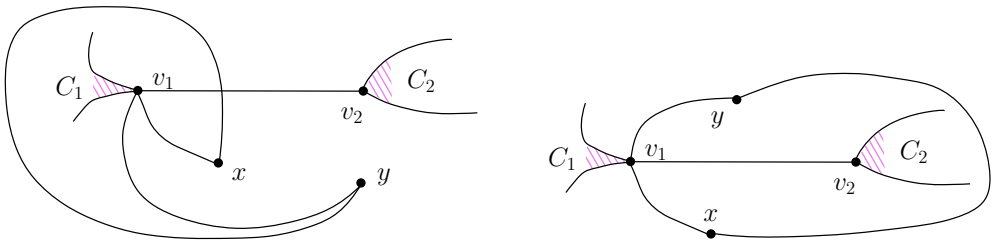


Figure 14 C_1 and C_2 are on different sides of Δv_1xy. Left: x and y belong to the same set of the bipartition. Right: x and y belong to different sets of the bipartition.

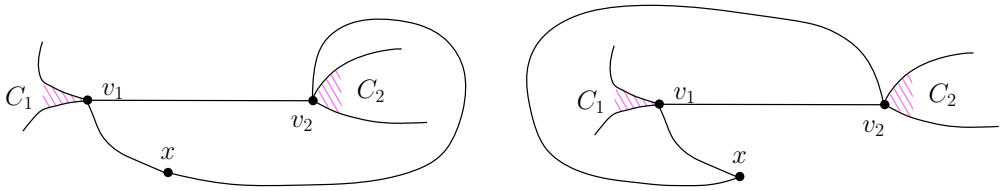


Figure 15 C_1 and C_2 are on different sides of Δv_1v_2x.

To prove that (i) holds, we take two vertices a and a' in A. (If A contains at most one vertex, (i) is vacuously true.) We use Lemma 17 to obtain the rotation of v_1 and v_2 and information on the bounding edges of C_1 and C_2. We conclude that the edge aa' crosses v_1v_2 because otherwise C_1 and C_2 would be on the same side of the triangle v_1aa' (see Figure 12(left) for an example) contradicting that C_1 and C_2 are antipodal. Thus, (i) follows and, by analogous arguments for B, (ii) follows.

To prove that (iii) holds for A and B, we take vertices $a \in A$ and $b \in B$. By Lemma 17, b is before C_1 and a is after C_1 around v_1. Thus, the edge ab cannot cross v_1v_2 so that C_1 and C_2 are on different sides of Δv_1ab (see Figure 12(right) for an example). Hence, (iii) follows. Finally, (iv)–(v) follow directly from the definitions of A and B.

To prove the other direction, we assume that R and D are a rotation system and drawing fulfilling (i)–(v), and show that R is generalized twisted. Let $\{v_2, b_1, \ldots, b_{n_1}, a_1, \ldots, a_{n_2}\}$ be the (clockwise) rotation of v_1 beginning at v_2, and $\{v_1, b'_1, \ldots, b'_{n_1}, a'_1, \ldots, a'_{n_2}\}$ be the

(clockwise) rotation of v_2 beginning at v_1. Let C_1 be the vi-cell of D defined by the edges $v_1b_{n_1}$ and v_1a_1, and C_2 be the vi-cell of D defined by the edges $v_2b'_{n_1}$ and $v_2a'_1$. We show that C_1 and C_2 are antipodal and thus R is generalized twisted by showing for all possible triangles in D that C_1 and C_2 are on different sides of that triangle. We analyze the triangles and show the following. If v_1 and v_2 are not incident to Δ, the number of crossings give that v_1 and v_2 – and thus C_1 and C_2 – are on different sides of Δ (see Figure 13). If v_1 or v_2 are incident to Δ, we use C_1 and C_2 being on different ends of v_1v_2 to show that C_1 and C_2 are on different sides of Δ (see Figures 14 and 15). ◀

5 Efficient algorithms

In this section, we turn to the algorithmic aspect and prove Theorem 3.

To check whether an abstract rotation system of K_n is generalized twisted, by Theorem 1, only the subrotation systems induced by five vertices need checking. This can be done in $O(n^5)$ time. (The only exception where subdrawings are not sufficient are abstract rotation systems of K_6, but for $n = 6$ all three generalized twisted rotation systems are known.) Alternatively, one can first check in $O(n^5)$ time whether the rotation system is realizable [2, 20]. If it is not, it cannot be generalized twisted. If it is, we can use the following faster algorithm for realizable rotation systems of K_n, which we developed using Theorem 2. The algorithm runs in $O(n^2)$ time and in two steps, which we describe in the remainder of this section.

Step 1. In the first step, the empty star triangles at an arbitrary vertex x are computed.

▶ **Lemma 18.** *Given a realizable rotation system R of K_n and a vertex x of R, the set of all empty star triangles at x is the same in each realization of R and can be computed from R in $O(n^2)$ time.*

Proof Sketch. By Lemma 5, a star triangle xyz at x is empty if and only if y and z are consecutive in the rotation of x. Hence, there are at most $n - 1$ empty star triangles at x. Whether a triangle xyz is a star triangle at x is determined by whether there exists a vertex w such that xw crosses yz. Since deciding for two edges if they cross each other can be done in constant time via the rotation system, deciding if a star triangle is empty requires $O(n)$ time. Thus, computing all empty star triangles at x can be done in $O(n^2)$ time. ◀

Step 2. In the second step, the previously computed empty star triangles at x and Theorem 2 are used to decide whether R is generalized twisted.

▶ **Lemma 19.** *Given a realizable rotation system R of K_n and the set of empty star triangles of a vertex x of R, it can be determined in $O(n^2)$ time whether R is generalized twisted.*

Proof Sketch. By Lemma 17, every drawing with generalized twisted rotation system has a pair (v_1, v_2) of compatible vertices. If (C_1, C_2) is a pair of antipodal vi-cells, then v_1 and v_2 lie on the boundary of C_1 and C_2 (or vice versa). We look for compatible vertices using empty triangles. Let x be the vertex with given empty star triangles. If there are not exactly two empty star triangles at x, then R is not generalized twisted by Lemma 6. Otherwise, we denote by $\Delta = xyz$ and $\Delta' = xy'z'$ the two empty star triangles at x. (Possibly, y or z coincides with y' or z'.) Using Lemma 6, we can show that if there is a compatible pair of vertices it must be in $S = \{(y, y'), (y, z'), (y, x), (z, y'), (z, z'), (z, x), (x, y'), (x, z')\}$. Checking compatibility of any pair in S by comparing their rotation systems takes linear time.

For every such potentially compatible pair (v_1, v_2), we check properties (i)–(v) of Theorem 2. Let $\{v = w_1, w_2, \ldots, w_{n-1}\}$ be the rotation of v_1. We define A and B as follows. If the rotation of v_2 is $\{v_1, w_{n-1}, \ldots, w_2\}$, then $B = \{w_2, \ldots, w_{n-1}\}$ and $A = \emptyset$ (or vice versa), and if the rotation of v_2 is $\{v_1, w_i, \ldots, w_2, w_{n-1}, \ldots, w_{i+1}\}$, for some i different from 1 and $n-1$, then $B = \{w_2, \ldots, w_i\}$ and $A = \{w_{i+1}, \ldots, w_{n-1}\}$. Then A and B clearly satisfy (iv) and (v) of Theorem 2. Checking if (i), (ii), and (iii) of Theorem 2 are satisfied requires $O(n^2)$ time.

Therefore, since $|S|$ is a constant, deciding if there is a pair (v_1, v_2) in S such that v_1 and v_2 are compatible and the properties (i)–(v) of Theorem 2 are satisfied for this pair can be done in $O(n^2)$ time. ◀

6 Conclusion and open problems

We presented two new characterizations for generalized twisted drawings based solely on their rotation systems. From these, we derived an $O(n^5)$-time algorithm that decides whether an abstract rotation system of K_n is generalized twisted and an $O(n^2)$-time algorithm to decide whether a realizable rotation system is generalized twisted. The obvious open question is whether the decision can also be done faster for abstract rotation systems. The quadratic time algorithm we give does not directly apply to them. There is a non-realizable rotation system of K_5 for which the output of the algorithm depends on the choice of vertex x and might be 'generalized twisted' or 'not generalized twisted'.

Generalized twisted drawings have been useful before, for example to find large plane substructures in simple drawings [5, 15]. Overcoming their rather geometrical definition, our new combinatorial characterizations might ease work with generalized twisted drawings and foster further results with this flavor. For example, Pach, Solymosi, and Tóth [21] showed that every simple drawing of a complete graph on n vertices contains $O(\log^{\frac{1}{8}} n)$ vertices inducing a complete subdrawing that is either convex or twisted. Recently, Suk and Zeng [25, 26] improved this result to $(\log n)^{\frac{1}{4}-o(1)}$ vertices. As both convex and twisted drawings contain large plane paths and cycles, this provides lower bounds on the size of plane paths and cycles that every simple drawing contains. It would be interesting to generalize this to show that every simple drawing contains a large generalized twisted or generalized convex subdrawing. Generalized convex drawings are – like generalized twisted drawings – a family of drawings of graphs that are well investigated [9, 11, 12]. Similarly to generalized twisted drawings, they admit characterizations via small constant sized subrotation systems [9]. Thus, a result guaranteeing the existence of a large generalized convex or generalized twisted drawing would help to extend our knowledge on simple drawings and, for example, to improve the lower bounds on the size of plane paths and cycles. We expect that this and several other questions on simple drawings can be tackled using the characterizations of this paper – independently or in combination. Especially, since the two given characterizations are very different from each other, with Theorem 1 being on all small subrotation systems and Theorem 2 being on two specific points and their rotation in the whole rotation system, they might foster the application of different other tools to obtain new results.

References

1 URL: https://figshare.com/projects/g-twisted_rotation_systems/184474.

2 Bernardo M. Ábrego, Oswin Aichholzer, Silvia Fernández-Merchant, Thomas Hackl, Jürgen Pammer, Alexander Pilz, Pedro Ramos, Gelasio Salazar, and Birgit Vogtenhuber. All good drawings of small complete graphs. In *Proc. 31st European Workshop on Computational*

Geometry EuroCG '15, pages 57–60, 2015. URL: http://www.ist.tu-graz.ac.at/files/publications/geometry/aafhpprsv-agdsc-15.pdf.

3 Bernardo M. Ábrego, Oswin Aichholzer, Silvia Fernández-Merchant, Pedro Ramos, and Gelasio Salazar. Shellable drawings and the cylindrical crossing number of K_n. *Discret. Comput. Geom.*, 52(4):743–753, 2014. doi:10.1007/S00454-014-9635-0.

4 Oswin Aichholzer, Alfredo García, Javier Tejel, Birgit Vogtenhuber, and Alexandra Weinberger. Twisted ways to find plane structures in simple drawings of complete graphs. In Xavier Goaoc and Michael Kerber, editors, *38th International Symposium on Computational Geometry, SoCG 2022*, volume 224 of *LIPIcs*, pages 5:1–5:18, 2022. doi:10.4230/LIPICS.SOCG.2022.5.

5 Oswin Aichholzer, Alfredo García, Javier Tejel, Birgit Vogtenhuber, and Alexandra Weinberger. Twisted ways to find plane structures in simple drawings of complete graphs. *Discret. Comput. Geom.*, 71(1):40–66, 2024. doi:10.1007/S00454-023-00610-0.

6 Oswin Aichholzer, Thomas Hackl, Alexander Pilz, Pedro Ramos, Vera Sacristán, and Birgit Vogtenhuber. Empty triangles in good drawings of the complete graph. *Graphs Comb.*, 31(2):335–345, 2015. doi:10.1007/S00373-015-1550-5.

7 Alan Arroyo, Dan McQuillan, R. Bruce Richter, and Gelasio Salazar. Drawings of k_n with the same rotation scheme are the same up to reidemeister moves (gioan's theorem). *Australas. J Comb.*, 67:131–144, 2017. URL: http://ajc.maths.uq.edu.au/pdf/67/ajc_v67_p131.pdf.

8 Alan Arroyo, Dan McQuillan, R. Bruce Richter, and Gelasio Salazar. Levi's lemma, pseudolinear drawings of K_n, and empty triangles. *J. Graph Theory*, 87(4):443–459, 2018. doi:10.1002/JGT.22167.

9 Alan Arroyo, Dan McQuillan, R. Bruce Richter, and Gelasio Salazar. Convex drawings of the complete graph: topology meets geometry. *Ars Math. Contemp.*, 22(3):3, 2022. doi:10.26493/1855-3974.2134.AC9.

10 Martin Balko, Radoslav Fulek, and Jan Kynčl. Crossing numbers and combinatorial characterization of monotone drawings of K_n. *Discret. Comput. Geom.*, 53(1):107–143, 2015. doi:10.1007/S00454-014-9644-Z.

11 Helena Bergold, Stefan Felsner, Meghana M. Reddy, Joachim Orthaber, and Manfred Scheucher. Plane Hamiltonian cycles in convex drawings. In *40th International Symposium on Computational Geometry, SoCG 2024, June 11-14, 2024, Athens, Greece*, volume 293 of *LIPIcs*, pages 18:1–18:16, 2024. doi:10.4230/LIPICS.SOCG.2024.18.

12 Helena Bergold, Joachim Orthaber, Manfred Scheucher, and Felix Schröder. Holes in convex and simple drawings. In Stefan Felsner and Karsten Klein, editors, *32nd International Symposium on Graph Drawing and Network Visualization, GD 2024*, volume 320 of *LIPIcs*, pages 5:1–5:9, 2024. doi:10.4230/LIPICS.GD.2024.5.

13 Stefan Felsner, Sandro Roch, and Manfred Scheucher. Arrangements of pseudocircles: On digons and triangles. In Patrizio Angelini and Reinhard von Hanxleden, editors, *30th International Symposium on Graph Drawing and Network Visualization, GD 2022*, volume 13764 of *Lecture Notes in Computer Science*, pages 441–455, 2022. doi:10.1007/978-3-031-22203-0_32.

14 Alfredo García, Javier Tejel, Birgit Vogtenhuber, and Alexandra Weinberger. Empty triangles in generalized twisted drawings of K_n. In Patrizio Angelini and Reinhard von Hanxleden, editors, *30th International Symposium on Graph Drawing and Network Visualization, GD 2022*, volume 13764, pages 40–48, 2022. doi:10.1007/978-3-031-22203-0_4.

15 Alfredo García, Javier Tejel, Birgit Vogtenhuber, and Alexandra Weinberger. Empty triangles in generalized twisted drawings of K_n. *J. Graph Algorithms Appl.*, 27(8):721–735, 2023. doi:10.7155/JGAA.00637.

16 Emeric Gioan. Complete graph drawings up to triangle mutations. In Dieter Kratsch, editor, *31st International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2005*, volume 3787 of *Lecture Notes in Computer Science*, pages 139–150, 2005. doi:10.1007/11604686_13.

17 Emeric Gioan. Complete graph drawings up to triangle mutations. *Discret. Comput. Geom.*, 67(4):985–1022, 2022. doi:10.1007/S00454-021-00339-8.

18 Heiko Harborth and Ingrid Mengersen. Drawings of the complete graph with maximum number of crossings. *Congressus Numerantium*, pages 225–225, 1992.

19 Jan Kynčl. Improved enumeration of simple topological graphs. *Discret. Comput. Geom.*, 50(3):727–770, 2013. doi:10.1007/S00454-013-9535-8.

20 Jan Kynčl. Simple realizability of complete abstract topological graphs simplified. *Discret. Comput. Geom.*, 64(1):1–27, 2020. doi:10.1007/S00454-020-00204-0.

21 János Pach, József Solymosi, and Géza Tóth. Unavoidable configurations in complete topological graphs. *Discrete Comput Geometry*, 30:311–320, 2003. doi:10.1007/s00454-003-0012-9.

22 Yan Alves Radtke, Stefan Felsner, Johannes Obenaus, Sandro Roch, Manfred Scheucher, and Birgit Vogtenhuber. Flip graph connectivity for arrangements of pseudolines and pseudocircles. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024*, pages 4849–4871, 2024. doi:10.1137/1.9781611977912.172.

23 Marcus Schaefer. Taking a detour; or, Gioan's theorem, and pseudolinear drawings of complete graphs. *Discret. Comput. Geom.*, 66(1):12–31, 2021. doi:10.1007/S00454-021-00296-2.

24 Marcus Schaefer, Jean Cardinal, and Tillmann Miltzow. The existential theory of the reals as a complexity class: A compendium, 2024. doi:10.48550/arXiv.2407.18006.

25 Andrew Suk and Ji Zeng. Unavoidable patterns in complete simple topological graphs. In Patrizio Angelini and Reinhard von Hanxleden, editors, *30th International Symposium on Graph Drawing and Network Visualization, GD 2022*, volume 13764 of *Lecture Notes in Computer Science*, pages 3–15, 2022. doi:10.1007/978-3-031-22203-0_1.

26 Andrew Suk and Ji Zeng. Unavoidable patterns in complete simple topological graphs. *Discret. Comput. Geom.*, 73(1):79–91, 2025. doi:10.1007/S00454-024-00658-6.

1-Planar Unit Distance Graphs with More Edges Than Matchstick Graphs

Eliška Červenková
Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Jan Kratochvíl
Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Abstract

Matchstick graphs are graphs that allow plane embedding with straight edges of equal length. One-planar unit distance graphs are graphs that allow a drawing in the plane in which all edges are straight-line segments of equal length and every edge crosses at most one other edge. The maximum number of edges of a matchstick graph (1-planar unit distance graph) of order n is denoted by $u_0(n)$ ($u_1(n)$, respectively). It is known that $u_0(n) = \lfloor 3n - \sqrt{12n-3} \rfloor$ holds for every n. At GD'24, Gehér and Tóth proved a slightly weaker upper bound on $u_1(n)$, but noted that no 1-planar unit distance graph G with more than $u_0(|V(G)|)$ vertices was known. They asked if $u_1(n) = u_0(n)$ holds for every n. We give a negative answer to this question in a much stronger way. We show that $u_1(n) > u_0(n)$ for every $n \geq 16135$. Furthermore, we show that the gap between $u_1(n)$ and $u_0(n)$ can be arbitrarily large by proving that for n large enough with respect to a constant $\alpha < \sqrt[4]{\frac{1}{3}}$, $u_1(n) - u_0(n) \geq \alpha \sqrt[4]{n}$.

2012 ACM Subject Classification Mathematics of computing → Graph theory

Keywords and phrases planar graph, unit distance graph, matchstick graph, 1-planar graph

Digital Object Identifier 10.4230/LIPIcs.GD.2025.26

Funding Both authors gratefully acknowledge support by grant No. 23-04949X of the Czech Science Foundation (GAČR).

Acknowledgements We thank the anonymous reviewers for valuable comments that helped increase the readability of the article.

1 Introduction

We consider unit distance geometric graphs, i.e., graphs drawn in the plane in such a way that edges are drawn as straight line segments of equal (unit) length. As usual, vertices are embedded as points, and no vertex may lie on an edge, unless being one of its end-vertices. We are interested in the following two subclasses of such graphs. *Matchstick graphs* are plane unit distance graphs, i.e., graphs that allow a unit distance embedding with no edge crossings. A more general class is formed by *1-planar unit distance graphs*, which allow unit distance drawings such that any edge is crossed by at most one other edge.

Matchstick graphs were introduced by Harborth in 1981 [4]. He conjectured that a matchstick graph on n vertices has at most $\lfloor 3n - \sqrt{12n-3} \rfloor$ edges. As an answer to the problem of Reutter [9], he actually proved his conjecture in the case when the distance of any two vertices of the graph is greater than or equal to 1 already in 1974 [3]. In the same paper, Harborth also showed that the bound is tight by constructing, for every positive integer n, a matchstick graph on n vertices that achieves this upper bound. However, the conjecture itself has only recently been proved by Lavollée and Swanepoel [8]. Although this answers completely the question of the maximum number of edges of a matchstick graph, there are many other open research questions. Quite a lot of attention has been paid to regular


© Eliška Červenková and Jan Kratochvíl;
licensed under Creative Commons License CC-BY 4.0



matchstick graphs [1], [5], [6], [7]. To complete the picture, let us mention the article by Salvia [10], which serves as a catalog of small examples, as well as the articles by Winkler [11] and by Winkler, Dinkelacker, and Vogel [12], [13], [14], [15], which provide an overview of the diversity of matchstick graphs.

Gehér and Tóth [2] considered 1-planar unit distance graphs. They observed an interesting phenomenon with regard to the maximum possible number of edges. Although general 1-planar n-vertex graphs (i.e., when the lengths of the edges are not required to be equal, and the edges need not be straight) may have as many as $4n-8$ edges compared to planar graphs having at most $3n-6$ edges, for unit distance graphs the coefficient of the linear term is the same. They showed that every 1-planar unit distance graph with n vertices has at most $3n-\frac{1}{10}\sqrt[4]{n}$ edges. At the same time, they pointed out that no construction of a 1-planar unit distance graph with more edges than a matchstick graph on the same number of vertices was known. They denoted by $u_0(n)$ ($u_1(n)$) the maximum number of edges of a matchstick (1-planar unit distance, respectively) graph on n vertices, and they have explicitly asked this open question.

Question [Problem 12 in [2]]. *Is it true that $u_0(n) = u_1(n)$ for every n?*

We first provide a negative answer to this question by exhibiting the graph depicted in Fig. 2 middle. This 1-planar unit distance graph has 31 vertices and 74 edges, while $u_0(31) = 73$. The main goal of this paper is to strengthen this answer by showing that if the number of vertices is large enough, 1-planar unit distance graphs always beat matchstick graphs in the number of edges.

▶ **Theorem 1.** *For every $n \geq 16135$, $u_1(n) > u_0(n)$.*

We further show that the gap between $u_0(n)$ and $u_1(n)$ widens as n tends to infinity.

▶ **Theorem 2.** *For any constant $\alpha < \sqrt[4]{\frac{1}{3}} = .7598\ldots$ and for every n sufficiently large (with respect to α), $u_1(n) - u_0(n) \geq \alpha\sqrt[4]{n}$.*

In the next sections, we prove these theorems using an explicit construction. In both cases, we first construct an infinite sequence of 1-planar unit distance graphs with large numbers of edges. These graphs arise as a combination of two patterns: a duplicated square grid in the core part of the graph, and two triangular grids attached to the top and bottom sides of the core. The sizes of the grids are carefully chosen to balance the trade-offs of the boundary effect of the two types of grids. The numbers of vertices of these graphs will form a sparse subset of the set of positive integers. In the second step, we show how to fill the gaps by fine-tuning the first construction by adding a few (some of them partial) layers to the triangular grids in order not to lose too much from the surplus of the number of edges before reaching the next graph in the main sequence.

2 The construction

2.1 Small cases

On $n = 6$ vertices, there are three non-isomorphic matchstick graphs with $u_0(6) = 9$ edges (see Fig. 1). There exists another non-isomorphic unit distance graph F depicted in Fig. 2 left. Note that this triangular prism graph is not a matchstick graph. However, it clearly is a 1-planar unit distance graph, as can be seen from drawing its vertices as points with coordinates $(0,0), (1,0), (0,1), (1,1), (\frac{1}{2}, \frac{\sqrt{3}}{2}), (\frac{1}{2}, 1+\frac{\sqrt{3}}{2})$. Concatenating five overlapping

copies of F and adding a top and a bottom path of equilateral triangles yields the ad hoc graph depicted in Fig. 2 middle. This is the smallest example of a 1-planar unit distance graph with more edges than a matchstick graph that we know of. It has 31 vertices and 74 edges, while $u_0(31) = \lfloor 93 - \sqrt{372 - 3} \rfloor = \lfloor 93 - 19.2... \rfloor = 73$.

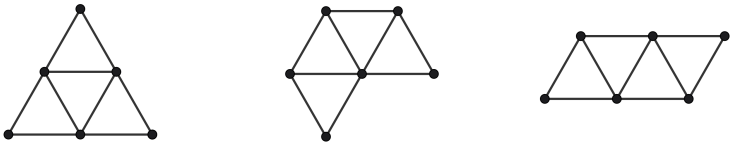


Figure 1 Three non-isomorphic matchstick graphs with 6 vertices and 9 edges.

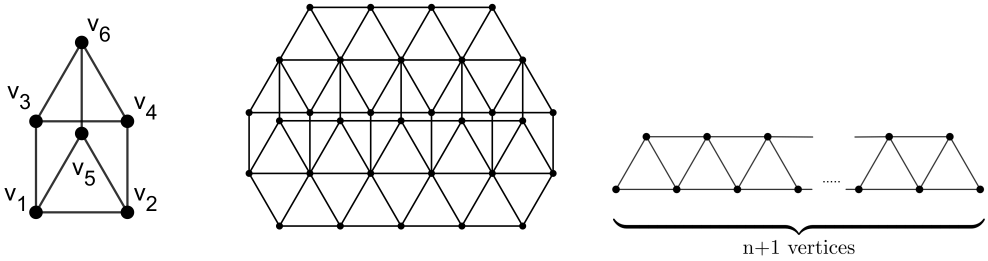


Figure 2 The triangular prism F (left), a 1-planar unit distance graph with 31 vertices and 74 edges (middle) and a path T_n of triangles (right).

2.2 Putting the prisms together

The graph from Fig. 2 middle contains all the ingredients of the general construction. The five copies of the prism graph F with the additional horizontal edges form a 1×5 square grid coupled with a shifted 1×4 one. A path T_4 of triangles is added on the top, and a path T_5 on the bottom (a *path of triangles* T_n is depicted in Fig. 2 right). The general construction is as follows. We build graphs $G_{t,k,a}$, parameterized by three parameters: k refers to the number of $k+1$ copies of the elementary building block F concatenated in a row, t is the number of such rows stacked on top of each other (to be precise, half of them stacked on top of each other, and the other half added below them in a reflection along the horizontal axis), and a is the number of paths of triangles stacked above (and below) the rows of F's. We will then tune the parameters k and a to achieve the best comparison with the maximum possible number of edges of a matchstick graph on the same number of vertices. We will show that the "well tuned" graphs "win" over $u_0(n)$ by t edges. Of course, the numbers of vertices given by this construction are very specific.

Let us now describe the construction in full detail. Note that t will always be even. To construct the graph $G_{t,k,a}$, begin with $k+1$ copies of F, denoted by $F^{1,i}$ for $i \in \{1, 2, \ldots, k+1\}$, with the corresponding vertex sets $\{v_j^{1,i} : j = 1, 2, \ldots, 6\}$. The larger graph $F^{(1\times k+1)}$ is formed by identifying the following vertices: $v_2^{1,i} = v_1^{1,i+1}$ and $v_4^{1,i} = v_3^{1,i+1}$ for $i \in \{1, 2 \ldots, k\}$. Additionally, for each $i \in \{1, 2, \ldots, k\}$, we include the edges $v_5^{1,i}v_5^{1,i+1}$ and $v_6^{1,i}v_6^{1,i+1}$.

All edges in $F^{(1\times k+1)}$ are represented by unit-length line segments. The only intersecting edges are of the form $v_2^{1,i}v_4^{1,i}$ and $v_5^{1,i}v_5^{1,i+1}$ for $i \in \{1, 2, \ldots, k\}$, and $v_5^{1,j}v_6^{1,j}$ and $v_3^{1,j}v_4^{1,j}$ for $j \in \{1, 2, \ldots, k+1\}$. Hence, $F^{(1\times k+1)}$ remains a 1-planar unit distance graph.

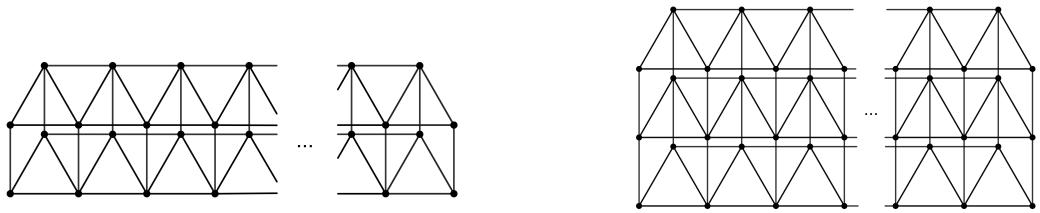


Figure 3 The graphs $F^{(1\times k+1)}$ (left) and $F^{(2\times k+1)}$ (right).

As previously stated, the graph F contains 6 vertices and 9 edges. Each additional copy of F contributes 4 new vertices and 10 edges because of the identifications made during the construction. Therefore, the graph $F^{(1\times k+1)}$ consists of $6+4k$ vertices and $9+10k$ edges.

In the next step, add a copy of $F^{(1\times k+1)}$ with corresponding vertices $v_j^{2,i}$ for $j \in \{1,\ldots,6\}, i \in \{1,2,\ldots,k+1\}$ and identify the vertices $v_1^{2,i} = v_3^{1,i}$, $v_2^{2,i} = v_4^{1,i}$, and $v_5^{2,i} = v_6^{1,i}$. Denote the resulting graph by $F^{(2\times k+1)}$. This procedure is referred to as *adding a row* to $F^{(1\times k+1)}$. It adds $3+2k$ vertices and $6+6k$ edges.

Repeat the above-described step $\left(\frac{t}{2}-1\right)$ times. Since each step adds a new row, for the resulting graph $F^{(\frac{t}{2}\times k+1)}$, it holds that

$$\begin{aligned} |V(F^{(\frac{t}{2}\times k+1)})| &= 6+4k+\left(\frac{t}{2}-1\right)(3+2k) = 3+2k+\frac{3t}{2}+kt, \\ |E(F^{(\frac{t}{2}\times k+1)})| &= 9+10k+\left(\frac{t}{2}-1\right)(6+6k) = 3+4k+3t+3kt. \end{aligned} \tag{1}$$

In the next step, attach a trapezoid consisting of a paths of triangles (the bottom one being T_k and the top one being T_{k-a+1}) to the graph $F^{(\frac{t}{2}\times k+1)}$ in such a way that the bottom side of the trapezoid coincides with the upper horizontal side of $F^{(\frac{t}{2}\times k+1)}$. Denote the resulting graph by $F_{t,k,a}$.

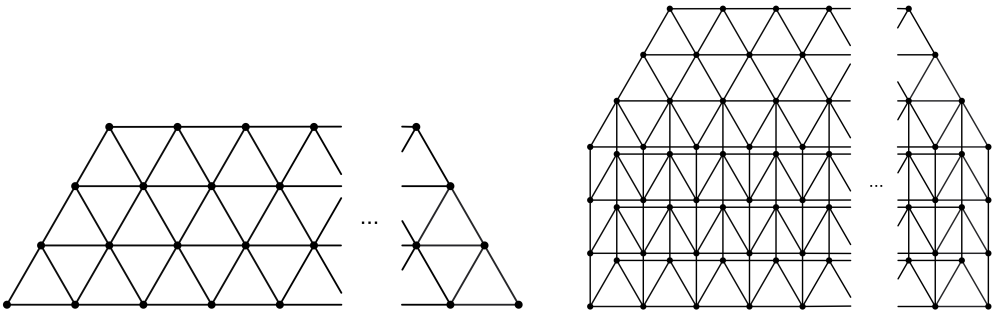


Figure 4 Illustration of a trapezoid consisting of 3 paths of triangles (left) and a graph $F_{t,k,a}$ for $t=6$ and $a=2$ (right).

It holds that

$$
\begin{aligned}
|V(F_{t,k,a})| &= |V(F^{(\frac{t}{2}\times k+1)})| + \sum_{\ell=k}^{k-a+1} \ell = 3 + 2k + \frac{3t}{2} + kt + \frac{(2k-a+1)a}{2} = \\
&= ka - \frac{(a-1)a}{2} + 3 + 2k + \frac{3t}{2} + kt, \\
|E(F_{t,k,a})| &= |E(F^{(\frac{t}{2}\times k+1)})| + \sum_{\ell=k}^{k-a+1} (3\ell - 1) = 3 + 4k + 3t + 3kt + 3\sum_{\ell=k}^{k-a+1} \ell - a = \\
&= 3 + 4k + 3t + 3kt + \frac{3(2k-a+1)a}{2} - a = \\
&= 3ak + \frac{-3a^2 + a}{2} + 3 + 4k + 3t + 3kt.
\end{aligned}
\tag{2}
$$

In the final step, the graph $G_{t,k,a}$ is obtained by reflecting $F_{t,k,a}$ across a horizontal axis and identifying the corresponding horizontal sides of length $k+1$.

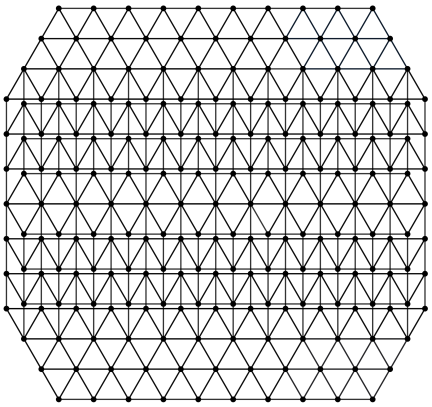


Figure 5 A graph $G_{6,11,2}$.

▶ **Proposition 3.** *For every three positive integers a, k and t, where t is even, $G_{t,k,a}$ is a 1-planar unit distance graph with $2ka - a^2 + a + 4 + 3k + 3t + 2tk$ vertices and $6ka - 3a^2 + a + 5 + 7k + 6t + 6kt$ edges.*

Proof. A 1-planar unit distance drawing of $G_{t,k,a}$ can be built along the lines of its definition. For the numbers of vertices and edges, note that $G_{t,k,a}$ is built by joining two copies of $F_{t,k,a}$ (the original and the flipped one) and these two copies share $k+2$ vertices and $k+1$ edges. Hence, $|V(G_{t,k,a})| = 2|V(F_{t,k,a})| - (k+2) = 2ka - a^2 + a + 4 + 3k + 3t + 2tk$ and $|E(G_{t,k,a})| = 2|E(F_{t,k,a})| - (k+1) = 6ka - 3a^2 + a + 5 + 7k + 6t + 6kt$. ◀

Finally, we choose a specific dependence of k on a and t, namely, $k = 2a + t + 1$, and set $H_{t,a} = G_{t,2a+t+1,a}$. It is not important for the main result, but we note that this choice optimizes the gain of $|E(G_{t,k,a})|$ over $u_0(|V(G_{t,k,n})|)$. This can be shown by standard methods in calculus. In the sequel, we set $n(t,a) = |V(H_{t,a})| = 3a^2 + 9a + 6at + 7 + 8t + 2t^2$. Note that $|E(H_{t,a})| = 9a^2 + 21a + 18at + 12 + 19t + 6t^2$.

▶ **Proposition 4.** *For every even positive integer t and every $a \geq t^2$, $|E(H_{t,a})| - u_0(n(t,a)) \geq t$.*

Proof. For $a = t^2$, we have

$$\begin{aligned} |V(H_{t,t^2})| &= n(t,t^2) = 3t^4 + 6t^3 + 11t^2 + 8t + 7, \\ |E(H_{t,t^2})| &= 9t^4 + 18t^3 + 27t^2 + 19t + 12 = 3n(t,t^2) - (6t^2 + 5t + 9) \end{aligned} \tag{3}$$

and

$$\begin{aligned} |E(H_{t,t^2})| - u_0(n(t,t^2)) &= 3n(t,t^2) - (6t^2 + 5t + 9) - \lfloor 3n(t,t^2) - \sqrt{12n(t,t^2) - 3} \rfloor \geq \\ &\geq 3n(t,t^2) - (6t^2 + 5t + 9) - (3n(t,t^2) - \sqrt{12n(t,t^2) - 3}) = \\ &= \sqrt{12n(t,t^2) - 3} - (6t^2 + 5t + 9) > t - 1. \end{aligned} \tag{4}$$

The last inequality follows from

$$\sqrt{12n(t,t^2) - 3} > 6t^2 + 6t + 8$$

which is equivalent to

$$\begin{aligned} 12n(t,t^2) - 3 &= 36t^4 + 72t^3 + 132t^2 + 96t + 81 > \\ &> 36t^4 + 72t^3 + (36 + 96)t^2 + 96t + 64 = (6t^2 + 6t + 8)^2. \end{aligned}$$

Denote

$$\begin{aligned} f_t(x) &= 9x^2 + 21x + 18xt + 12 + 19t + 6t^2 - \\ &\quad - (3(3x^2 + 9x + 6xt + 7 + 8t + 2t^2) - \sqrt{12(3x^2 + 9x + 6xt + 7 + 8t + 2t^2) - 3}) = \\ &= \sqrt{12(3x^2 + 9x + 6xt + 7 + 8t + 2t^2) - 3} - (6x + 5t + 9) \end{aligned}$$

for a real variable x. Note that $|E(H_{t,a})| - u_0(n(t,a)) \geq f_t(a)$ for every positive integer a. We claim that $f_t(x)$ is increasing for $x > 0$ for every $t > 0$. This follows from the first derivative in x. A simple calculation shows that

$$\frac{df_t}{dx} = -6 + \frac{1}{2} \cdot \frac{72x + 108 + 72t}{\sqrt{36x^2 + 108x + 72xt + 81 + 96t + 24t^2}} > 0 \tag{5}$$

is equivalent to $12t^2 + 12t > 0$, which is true for all $t > 0$.

Since $f_t(x)$ is increasing and $f_t(t^2) > t - 1$ as we have seen above, we have

$$|E(H_{t,a})| - u_0(n(t,a)) \geq f_t(a) > t - 1 \tag{6}$$

for every $a \geq t^2$. And since $|E(H_{t,a})| - u_0(n(t,a))$ is an integer, we get

$$|E(H_{t,a})| - u_0(n(t,a)) \geq t \tag{7}$$

for every $a \geq t^2$. ◀

▶ **Corollary 5.** *For every integer t, there are infinitely many values n such that $u_1(n) - u_0(n) \geq t$. Hence $\limsup_{n\to\infty}(u_1(n) - u_0(n)) = \infty$.*

2.3 Proof of Theorem 1

In this subsection, we will show how to fill the gaps between consecutive $n(t,a)$ and $n(t,a+1)$ for which we know, by Proposition 4, that $u_1(n(t,a)) > u_0(n(t,a))$ and $u_1(n(t,a+1)) > u_0(n(t,a+1))$. For each $n \in \{n(t,a)+1, \ldots, n(t,a+1)-1\}$, we will construct a graph $L_{t,n}$ by adding $n - n(t,a)$ vertices to $H_{t,a}$ (see the detailed description below). The following observation will be quite useful to argue about the gain of $L_{t,n}$ over $u_0(n)$.

▶ **Lemma 6.** *Let $G_0, G_1, \ldots, G_n$ be a sequence of graphs such that for every $i = 1, \ldots, n$, G_i is obtained from G_{i-1} by adding a vertex x_i of degree 2 or 3. Then $|E(G_n)| - u_0(|V(G_n)|) \geq |E(G_0)| - u_0(|V(G_0)|) - k$, where $k = |\{i : deg_{G_i} x_i = 2\}|$.*

Proof. Let us denote by $n_i = n_0 + i$ the number of vertices of G_i, $i = 0, 1, \ldots, n$. Observe that

$$u_0(n_i) = \lfloor 3n_0 + 3i - \sqrt{12n_i - 3} \rfloor \leq \lfloor 3n_0 + 3i - \sqrt{12n_0 - 3} \rfloor \leq u_0(n_0) + 3i.$$

Adding the vertex x_i to G_{i-1} adds 2 or 3 edges, depending on $\deg_{G_i} x_i$. Thus,

$$|E(G_n)| = |E(G_0)| + 3n - k$$

for $k = |\{i : \deg_{G_i} x_i = 2\}|$. Putting these two together, we obtain

$$|E(G_n)| - u_0(n_n) = |E(G_0)| + 3n - k - u_0(n_n) \geq$$

$$\geq |E(G_0)| + 3n - k - u_0(n_0) - 3n = |E(G_0)| - u_0(n_0) - k. \quad \blacktriangleleft$$

Proof of Theorem 1. Fix an even integer $t, t \geq 8$ and examine two consecutive graphs $H_{t,a}$ and $H_{t,a+1}$. For $n \in \{|V(H_{t,a})|, |V(H_{t,a+1})|\}$, our construction guarantees a graph G on n vertices such that $|E(G)| - u_0(n) \geq t$. Since the goal is to extend the construction to cover values of n strictly between $|V(H_{t,a})|$ and $|V(H_{t,a+1})|$, we recall the number of vertices in both graphs explicitly:

$$\begin{aligned} |V(H_{t,a})| &= 3a^2 + 9a + 6at + 7 + 8t + 2t^2, \\ |V(H_{t,a+1})| &= 3a^2 + 15a + 19 + 6at + 14t + 2t^2. \end{aligned} \tag{8}$$

The number of vertices that separate $H_{t,a}$ from $H_{t,a+1}$ is

$$\begin{aligned} |V(H_{t,a+1})| - |V(H_{t,a})| &= 3a^2 + 15a + 19 + 6at + 14t + 2t^2 - \\ - (3a^2 + 9a + 6at + 7 + 8t + 2t^2) &= 6a + 6t + 12. \end{aligned} \tag{9}$$

The geometric structure of $H_{t,a}$, which has a perimeter of the shape of an octagon with horizontal sides of length $a + 1 + t$, will be further exploited. Attaching a path of triangles T_ℓ to a horizontal side of length ℓ contributes ℓ vertices and $3\ell - 1$ edges. This implies that by adding these vertices one by one, we add 1 vertex of degree 2 and then $\ell - 1$ vertices of degree 3. By Lemma 6, such an addition of vertices decreases the edge surplus by at most one.

Suppose further that $a \geq t^2$. Then $a \geq t^2 > 2t$, as $t \geq 8$ is assumed. Then attach paths of triangles T_{a+1+t}, T_{a+t} and T_{a+t-1} to the upper horizontal side of $H_{t,a}$, and T_{a+1+t}, T_{a+t}, T_{a+t-1} and T_{a+t-2} to the bottom horizontal side of $H_{t,a}$. This operation adds:

$$\begin{aligned} 2(a + 1 + t + (a + t) + (a + t - 1)) + (a + t - 2) &= 7a - 2 + 7t = \\ = 6a + a - 2 + 6t + t \geq 6a - 2 + 6t + 3t &> 6a + 6t + 13 \end{aligned}$$

vertices, which is more than enough to cover the gap computed in (9).

Assume that an integer $n \in \{|V(H_{t,a})|, \ldots, |V(H_{t,a+1})|\}$ is given. Construct the graph $L_{t,n}$ by adding the rows attached to $H_{t,a}$ from above and below, until the number of vertices is n (the last row added may remain unfinished). The resulting graph is shown in Fig. 6.

In total, we have added at most 7 vertices of degree 2, the remaining ones were all of degree 3. Lemma 6 implies that

$$|E(L_{t,n})| - u_0(n) \geq |E(H_{t,a})| - u_0(|V(H_{t,a})|) - 7 \geq t - 7 \geq 1 \tag{10}$$

as $t \geq 8$ is assumed.

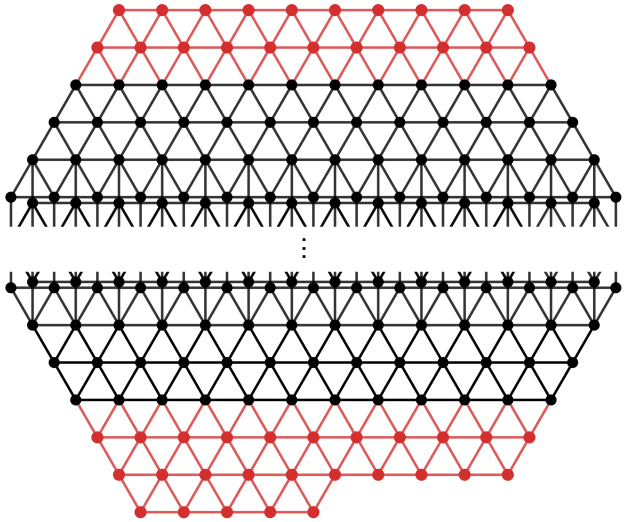


Figure 6 A construction of a graph $L_{t,n}$.

This implies that for every even integer $t, t \geq 8$, for all $n \geq |V(H_{t,t^2})|$, there exists a graph $L_{t,n}$ on n vertices satisfying

$$|E(L_{t,n})| - u_0(n) \geq 1.$$

Substituting the smallest values for a and t, which means $t = 8$ and $a = 64$, we get

$$|V(H_{8,64})| = 3 \cdot 64^2 + 9 \cdot 64 + 6 \cdot 64 \cdot 8 + 7 + 8 \cdot 8 + 2 \cdot 8^2 = 16135.$$

Therefore, for all $n \geq 16135$, there exists a 1-planar unit distance graph on n vertices with more edges than any matchstick graph on the same number of vertices has. ◀

The proof has the following useful corollary.

▶ **Corollary 7.** *For every even positive integer $t \geq 8$ and for every $n \geq n(t, t^2)$, $u_1(n) - u_0(n) \geq t - 7$.*

Proof. If $n \geq n(t, t^2)$, then $n \in \{n(t, a), \ldots, n(t, a + 1) - 1\}$ for some $a \geq t^2$. Then (10) implies that $u_1(n) - u_0(n) \geq t - 7$. ◀

2.4 Proof of Theorem 2

In this subsection, we show a lower bound on the growth rate of the surplus $u_1(n) - u_0(n)$ as n tends to infinity. We utilize the construction from the previous subsection. Again, we first show that the bound from Theorem 2 holds for infinitely many isolated values of n, and then fill the gaps. Let $\alpha < \sqrt[4]{\frac{1}{3}}$ be an arbitrary positive constant from the statement of the theorem and let β be another constant, $\alpha < \beta < \sqrt[4]{\frac{1}{3}}$. The constants α and β stay fixed for the rest of this subsection. We use the notation $n(t, a) = |V(H_{t,a})| = 3a^2 + 9a + 6at + 7 + 8t + 2t^2$ which was introduced in Subsection 2.2.

▶ **Proposition 8.** *For sufficiently large t, we have $t \geq \beta \sqrt[4]{n(t, t^2)}$.*

Proof. Extend the function $n(t,t^2) = 3t^4+6t^3+11t^2+8t+7$, which has so far been defined only for even positive integers t, to all positive real values of t. For $n = n(t,t^2)$, Wolfram Alpha tells us (and it can then be easily checked by hand) that t can be expressed as

$$t = \frac{1}{6}\left(\sqrt{12\sqrt{3n-5}-39}-3\right).$$

It follows that

$$t = \sqrt{\sqrt{\frac{n}{3}-\frac{5}{9}}-\frac{39}{36}}-\frac{1}{2} \geq \beta\sqrt[4]{n}$$

holds for sufficiently large n (and hence also for sufficiently large t), as long as $\beta < \sqrt[4]{\frac{1}{3}}$. ◀

Proof of Theorem 2. Let $t_0 \geq 8$ be an even integer large enough, so that

$$t_0 - 7 \geq \frac{\alpha}{\beta}(t_0+2) \text{ and } t_0 \geq \beta\sqrt[4]{n(t_0,t_0^2)}.$$

(Since the involved functions are monotone, it follows that

$$t - 7 \geq \frac{\alpha}{\beta}(t+2) \text{ and } t \geq \beta\sqrt[4]{n(t,t^2)}$$

hold for every $t \geq t_0$.)

For every $n \geq n(t_0,t_0^2)$, there is an even integer t such that $n(t,t^2) \leq n < n(t+2,(t+2)^2)$. Then, using this t, it follows from Corollary 7 and Proposition 8 that

$$u_1(n) - u_0(n) \geq t - 7 \geq \frac{\alpha}{\beta}(t+2) \geq \frac{\alpha}{\beta}\beta\sqrt[4]{n(t+2,(t+2)^2)} > \alpha\sqrt[4]{n}.$$ ◀

3 Conclusion

We have answered in negative the question of Gehér and Tóth, who asked if $u_1(n) = u_0(n)$ for every n by showing that for every large enough n, the difference $u_1(n) - u_0(n)$ is at least as large as $\alpha\sqrt[4]{n}$ for any constant $\alpha < \sqrt[4]{\frac{1}{3}}$. It is interesting that a function of the same growth rate appears in the upper bound for $u_1(n)$ proven by Gehér and Tóth. In fact, their result can be reformulated as $3n - u_1(n) \geq \frac{\sqrt[4]{n}}{10}$. Is this a coincidence?

All our examples of 1-planar unit distance graphs are non-planar. This invokes the following open problem (which seems to have been asked during the discussion at Graph Drawing '24).

Problem 1. Does there exist a planar graph which has a 1-planar unit distance drawing and which has more edges than any matchstick graph on the same number of vertices?

Also all our examples allow for 1-planar unit distance drawings in which only vertical and horizontal edges cross. Define $u_{0.5}(n)$ as the maximum number of edges of such an n-vertex graph.

Problem 2. Prove an upper bound on $u_{0.5}(n)$ better than $3n - c\sqrt[4]{n}$.

Problem 3. Is $u_{0.5}(n) < u_1(n)$ for some n? For infinitely many n's?

References

1 Aart Blokhuis. Regular finite planar maps with equal edges, 2019. doi:10.48550/arXiv.1401.1799.

2 Panna Gehér and Géza Tóth. 1-Planar Unit Distance Graphs. In Stefan Felsner and Karsten Klein, editors, *32nd International Symposium on Graph Drawing and Network Visualization (GD 2024)*, volume 320 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:9. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.GD.2024.6.

3 Heiko Harborth. Lösung zu Problem 664A. *Elem. Math.*, 29:14–15, 1974.

4 Heiko Harborth. Point sets with equal numbers of unit-distance neighbors (Abstract). In *Discrete Geometry*, Oberwolfach, Tagungsbericht 31/1981, pages 11–12. Mathematisches Forschungsinstitut Oberwolfach, 1981.

5 Sascha Kurz. A lower bound for 4-regular planar unit distance graphs. *Geombinatorics Quarterly*, 21:63–72, 2011. doi:10.48550/arXiv.1401.4377.

6 Sascha Kurz and Giuseppe Mazzuoccolo. 3-regular matchstick graphs with given girth. *Geombinatorics Quarterly*, 19:156–173, 2009. doi:10.48550/arXiv.1401.4360.

7 Sascha Kurz and Rom Pinchasi. Regular matchstick graphs. *The American Mathematical Monthly*, 118:264–267, 2011. doi:10.4169/amer.math.monthly.118.03.264.

8 Jérémy Lavollée and Konrad Swanepoel. A tight bound for the number of edges of matchstick graphs. *Discrete Computational Geometry*, 72:1530–1544, 2023. doi:10.1007/s00454-023-00530-z.

9 O. Reutter. Problem 664A. *Elem. Math.*, 27:19, 1972.

10 Raffaele Salvia. A catalog of matchstick graphs, 2015. doi:10.48550/arXiv.1303.5965.

11 Mike Winkler. A catalog of 4-regular matchstick graphs with 63 - 70 vertices, 2017. doi:10.48550/arXiv.1705.04715.

12 Mike Winkler, Peter Dinkelacker, and Stefan Vogel. Minimal completely asymmetric (4; n)-regular matchstick graphs, 2016. doi:10.48550/arXiv.1609.06972.

13 Mike Winkler, Peter Dinkelacker, and Stefan Vogel. New minimal (4; n)-regular matchstick graphs. *Geombinatorics Quarterly*, 27:26–44, 2017. doi:10.48550/arXiv.1604.07134.

14 Mike Winkler, Peter Dinkelacker, and Stefan Vogel. On the existence of 4-regular matchstick graphs, 2017. doi:10.48550/arXiv.1705.00293.

15 Mike Winkler, Peter Dinkelacker, and Stefan Vogel. A 3-regular matchstick graph of girth 5 consisting of 54 vertices. *Geombinatorics Quarterly*, 29:116–121, 2019. doi:10.48550/arXiv.1903.04304.

Geometry Matters in Planar Storyplans

Alexander Dobler
TU Wien, Austria

Maximilian Holzmüller
TU Wien, Austria

Martin Nöllenburg
TU Wien, Austria

Abstract

A *storyplan* visualizes a graph $G = (V, E)$ as a sequence of ℓ frames $\Gamma_1, \ldots, \Gamma_\ell$, each of which is a drawing of the induced subgraph $G[V_i]$ of a vertex subset $V_i \subseteq V$. Moreover, each vertex $v \in V$ is contained in a single consecutive sequence of frames $\Gamma_i, \ldots, \Gamma_j$, all vertices and edges contained in consecutive frames are drawn identically, and the union of all frames is a drawing of G. In GD 2022, the concept of *planar storyplans* was introduced, in which each frame must be a planar (topological) drawing. Several (parameterized) complexity results for recognizing graphs that admit a planar storyplan were provided, including NP-hardness. In this paper, we investigate an open question posed in the GD paper and show that the geometric and topological settings of the planar storyplan problem differ: We provide an instance of a graph that admits a planar storyplan, but no planar *geometric* storyplan, in which each frame is a planar straight-line drawing. Still, by adapting the reduction proof from the topological to the geometric setting, we show that recognizing the graphs that admit planar geometric storyplans remains NP-hard.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Graph theory; Human-centered computing → Graph drawings

Keywords and phrases geometric storyplan, planarity, straight-line drawing, dynamic graph drawing

Digital Object Identifier 10.4230/LIPIcs.GD.2025.27

Related Version *Full Version*: `https://arxiv.org/abs/2508.12747` [13]
Based on Maximilian Holzmüller's master thesis: `https://repositum.tuwien.at/handle/20.500.12708/213348` [16]

Funding *Alexander Dobler*: Vienna Science and Technology Fund (WWTF) grant [10.47379/ICT19035].
Martin Nöllenburg: Vienna Science and Technology Fund (WWTF) grant [10.47379/ICT19035].

1 Introduction

Planar storyplans, introduced in 2022 [4], represent an approach to draw possibly dense and non-planar graphs as a sequence of $\ell \geq 1$ planar subdrawings called frames. In addition to the planarity condition for each frame, their sequence must satisfy certain visual consistency criteria in order to be a valid planar storyplan: (i) each vertex belongs to a single, non-empty interval of frames, (ii) each frame shows the subgraph induced by the vertices of the frame, (iii) the shared vertices and edges of any two consecutive frames have the same geometric representation, and (iv) every edge must be visible in at least one frame, i.e., the incident vertices of every edge must co-occur in some frame. Storyplans belong to the class of gradual graph visualizations, which depict a graph in a sequential story-like or unordered small-multiples fashion. The goal of such gradual visualizations is to show a complex graph G as a collection of simpler drawings, e.g., planar drawings of subgraphs whose union represents G. Identical vertices and edges in the individual subdrawings can be mentally linked by requiring that they are always drawn the same if they occur in multiple


© Alexander Dobler, Maximilian Holzmüller, and Martin Nöllenburg;
licensed under Creative Commons License CC-BY 4.0



subdrawings. Such visualizations find applications both for visualizing dynamic graphs [1,17], in which vertices and edges occur over time, and for graphs that are decomposed into multiple simpler subgraphs without a sequential order [11,12]. Storyplans with their temporal sequence of frames are related to drawing graphs in a streaming model. In the storyplan problem, however, one can choose the sequence in which vertices enter and leave the story, whereas this sequence is part of the input in streaming models [2,3,8,15] and in graph stories [6,7,9,10].

The authors of [4,5] investigated the complexity of deciding whether a given graph admits a planar storyplan in a topological setting, with edges drawn as simple curves between their endpoints. They showed NP-completeness and fixed-parameter tractability parameterized by the vertex cover number or the feedback edge set number of the graph. Furthermore, they proved that every graph of treewidth at most 3 admits a planar storyplan, and that such a planar storyplan (actually with straight-line edges) can be computed in linear time. In [14] the investigation of storyplans was extended to outerplanar and forest storyplans, in which each frame is not just planar, but actually outerplanar or a forest. They proved a strict containment relation between forest, outerplanar, and planar storyplans and identified graph families that do or do not always admit these storyplans. In the affirmative case, their storyplans use straight-line edges.

In this paper, we investigate an open question by several authors [4, 5] and explore the differences between planar (topological) storyplans and planar geometric storyplans. Obviously, every graph that admits a planar geometric storyplan also admits a planar storyplan. We prove that the converse is not true by providing a counterexample of a graph with 28 vertices that admits a planar storyplan, but none with straight-line edges. The main challenge in our geometric construction is to enforce an obstruction to straight-line visibility for any possible placement of the vertices and for any possible vertex-to-frame assignment. As a second result, we adapt the hardness proof of from [4, 5] to establish that deciding if a given graph admits a planar geometric storyplan remains NP-hard.

2 Preliminaries

For $n \in \mathbb{N}$, we use $[n] = \{1, 2, \ldots, n\}$ as a shorthand. For $a \leq b \in \mathbb{N}$, we define $[a, b] = \{a, a+1, \ldots, b\}$. For a graph G, let $V(G)$ be its vertex set and let $E(G)$ be its edge set. For $V' \subseteq V(G)$, $G[V']$ is the subgraph of G induced by V'. Given an induced subgraph G' of some graph G, and a vertex of G not contained in $V(G')$, $G' + v$ is defined as $G[V(G') \cup \{v\}]$. We denote by uv the edge $\{u, v\}$ connecting vertices u and v.

The storyplan problem. A *storyplan* of a graph G on time steps $[\ell]$, $\ell \in \mathbb{N}$, consists of a pair $(A, \mathcal{D})$. For each $v \in V(G)$, $A(v) = [s_v, e_v] \subseteq [\ell]$ is the *visible interval* of v. We say that v *appears* at s_v, is *visible at* each $s_v, s_v + 1, \ldots, e_v$, and *disappears* after e_v. It must hold for a pair of adjacent vertices $u, v \in V$ that $A(u) \cap A(v) \neq \emptyset$, i.e., there is a time step in which the edge uv can be drawn. We say in this case that u and v *co-occur*. For each $t \in [\ell]$, $G[t] = G[\{v \in V \mid t \in A(v)\}]$ is the *frame graph at* t. For a subgraph G' of G, we say that G' is *visible at* t if all of its vertices are visible. The *drawing function* $\mathcal{D}$ represents a drawing $\mathcal{D}(v)$ of each vertex $v \in V(G)$ and drawing $\mathcal{D}(e)$ of each edge $e \in E(G)$. Further, for each subgraph G' of G, $\mathcal{D}(G')$ is the drawing of G', i.e., the drawing function is applied to each vertex and edge of G'. For each $t \in [\ell]$, we call $\Gamma_t := \mathcal{D}(G[t])$ the *frame at* t. For a storyplan, it must hold that each frame represents a simple drawing of the corresponding induced subgraph. A storyplan is *planar* if each frame corresponds to a planar drawing. A storyplan is *geometric* (or *straight-line*) planar if each frame corresponds to a planar drawing such that each edge is represented by a straight-line segment.

We consider the following two problems.

▶ **Problem 1.** *Given a graph G, does G admit a planar storyplan?*

▶ **Problem 2.** *Given a graph G, does G admit a planar geometric storyplan?*

Note that these definitions are slightly different from those in [5]. There, each vertex appears in a unique frame, while we permit multiple vertices appearing in a single frame. It is easy to see, though, that these two definitions are equivalent with regard to the existence of a planar (geometric) storyplan for a given graph G.

3 Geometry Matters in Planar Storyplans

In this section, we show our following main result.

▶ **Theorem 1.** *There is a graph that admits a planar storyplan and that does not admit a planar geometric storyplan.*

The graph G is defined as follows. It consists of three four-cycles A, B, and C with vertices a_i, b_i, c_i, $i \in [4]$ and edges $e_i^A = a_i a_{i \bmod 4+1}$, $e_i^B = b_i b_{i \bmod 4+1}$, and $e_i^C = c_i c_{i \bmod 4+1}$ for $i \in [4]$. Furthermore, the graph contains eight *apex vertices* q_i^j, $i \in [4], j \in [2]$, such that each q_i^j is connected to all vertices of A, B, and C. Lastly, eight more *edge vertices* r_i^j, $i \in [4], j \in [2]$ are added. Each r_i^j is connected to q_i^j and to all vertices from e_i^A, e_i^B, and e_i^C.

We define $Q = \{q_i^j \mid i \in [4], j \in [2]\}$ and $R = \{r_i^j \mid i \in [4], j \in [2]\}$. For each $i \in [4]$ and $j \in [2]$, $P_i^j = \{q_i^j, r_i^j\}$ is an *apex-pair*.

Below, we give a formal definition of G. Further, the graph can also be parsed from one of its planar storyplans in Figure 1.

$$
\begin{aligned}
V(G) =& \{a_i, b_i, c_i \mid i \in [4]\} \cup \{q_i^j, r_i^j \mid i \in [4], j \in [2]\} \\
E(G) =& \{a_i a_{i \bmod 4+1}, b_i b_{i \bmod 4+1}, c_i c_{i \bmod 4+1} \mid i \in [4]\} \cup \{r_i^j q_i^j \mid i \in [4], j \in [2]\} \cup \\
& \{r_i^j a_i, r_i^j a_{i \bmod 4+1}, r_i b_i, r_i b_{i \bmod 4+1}, r_i^j c_i, r_i^j c_{i \bmod 4+1} \mid i \in [4], j \in [2]\} \cup \\
& \{q_i^j a_k, q_i^j b_k, q_i^j c_k \mid i, k \in [4], j \in [2]\}
\end{aligned}
$$

We prove both statements of Theorem 1 in the following two sections.

3.1 G Has a Planar Storyplan

A planar storyplan of G with eight frames $\Gamma_1, \ldots, \Gamma_8$ is given in Figure 1. The four-cycles A, B, C are visible in all frames. Each apex-pair q_i^j, r_i^j for $i \in [4]$ and $j \in [2]$ is visible only in frame $\Gamma_{(j-1)4+i}$. As the neighborhoods of apex pairs q_i^1, r_i^1 and q_i^2, r_i^2, $i \in [4]$, w.r.t. the four-cycles are the same, they can be represented by the same drawing. This is why we show only four drawings in the figure.

3.2 G Does Not Have a Planar Geometric Storyplan

This section serves the purpose of showing the following.

▶ **Theorem 2.** *The graph G does not admit a planar geometric storyplan.*

We proceed to assume that G admits a planar geometric storyplan $(A, \mathcal{D})$, and we will arrive at a contradiction. We require a sequence of lemmas that will lead to this contradiction.

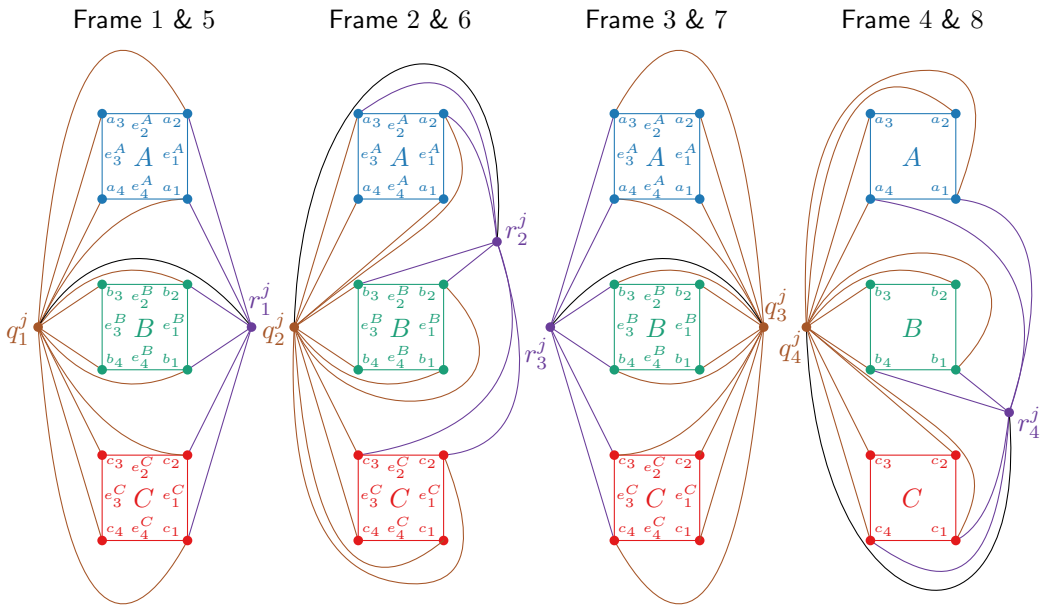


Figure 1 A planar storyplan for the graph G with eight frames. Frames k and $k+4$ show the same drawing for each $k \in [4]$. For frames $1, 2, 3, 4$, $j = 1$; for frames $5, 6, 7, 8$, $j = 2$.

▶ **Lemma 3.** *Let $I = \{A(q_i^j) \mid q_i^j \in Q\}$ be the visible intervals of the vertices in Q. Then there exists a set $I' \subset I$ of cardinality at least 6, such that for all $[s, e] \in I'$ and all $t \in [s, e]$, A, B, and C are visible at t.*

Proof. First, there is a time step in which all cycle vertices from a single cycle, say A, are visible. Otherwise, consider the time step e_A after which the first cycle vertex of A, say a_1, disappears, yet a_3 has not yet appeared. Then, in e_A, the three cycle vertices a_1, a_2, a_4 of A and all eight apex vertices must be visible since they need to see both a_1 and a_3; they induce a non-planar $K_{3,8}$, contradicting that every frame is planar. This holds analogously for the cycles B, C. Secondly, there is a time step in which all cycle vertices of all three cycles are visible. Otherwise, consider the time step after which the first cycle vertex disappears. Again, at this time step, we have at least a non-planar $K_{4,8}$ by the same argumentation that the apex vertices need to see both the disappearing cycle vertex and at least one cycle vertex that has not yet appeared.

Now sort the intervals in I by their start points. Consider the first two intervals $[s, e], [s', e']$ in this order. Let q and q' be their respective apex-vertices. We show that $[s, e] \cap [s', e'] = \emptyset$. By our previous arguments, there is a time step $t \in [s, e]$ in which q and all cycle vertices are visible. If $s \leq s' \leq e$, then there is furthermore a time step t' with $s' \leq t' \leq e$, such that q, q', and all cycle vertices are visible. Notice that this amounts to 14 vertices and 36 edges. Further, the induced drawing of the four-cycles and the two apex-vertices must clearly have a face incident to four vertices – four vertices of one of the four-cycles. Adding a chord, keeping the structure planar, we would end up with 14 vertices and 37 edges, a contradiction to Euler's formula. Symmetrically, by considering the last two intervals when sorted by their end points, we also see that their intersection is empty. The statement follows directly, as each apex must co-occur with each cycle-vertex. ◀

▶ **Corollary 4.** *There exist at least 6 frames, each containing a different apex pair and all four-cycles.*

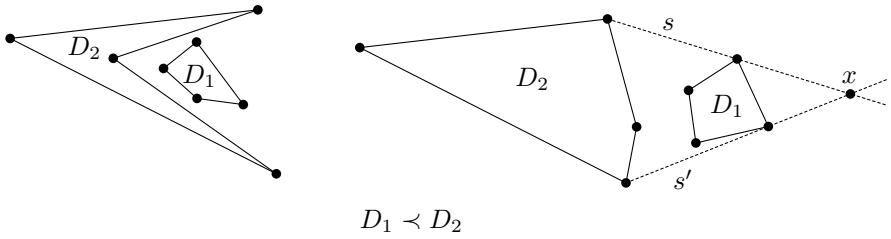


Figure 2 The two cases of a $D_1 \prec D_2$. In the second case, the point x is closer to D_1 than to D_2.

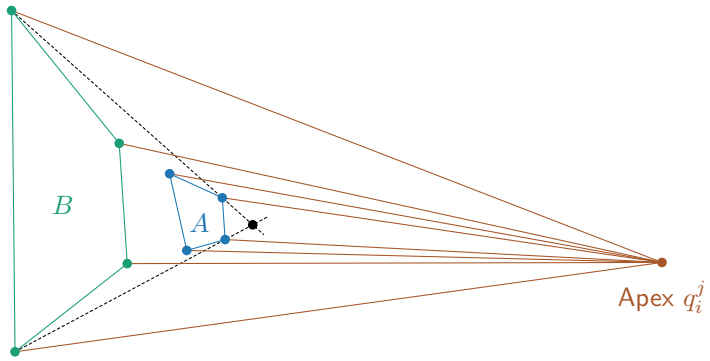


Figure 3 $\mathcal{D}(A) \prec \mathcal{D}(B)$ is implied by A lying inside of one of the faces of the drawing of $A + q_i^j$.

▶ **Lemma 5.** *There exists a pair from $\{A, B, C\}$ that is not nested, i.e., in the drawing of only these two four-cycles, the outer face is incident to all vertices.*

Proof. This follows directly from Lemma 3, as there exists a frame with some apex vertex and all three four-cycles. Clearly, the statement holds, as there is otherwise no face (in the induced drawing of A, B, and C) that contains all four-cycle vertices. ◀

In the remainder of the proof, we assume that A and B form a non-nested pair.

Next, we want to define one of the drawings of A or B to be "smaller", see Figure 2. This will be useful in showing that for one of the four-cycles A and B, in the drawing of this four-cycle plus an apex-vertex, the other four-cycle will always lie on the outer face.

▶ **Definition 6.** *Consider two simple quadrilaterals D_1 and D_2, such that the outer face is incident to all eight vertices. Define $D_1 \prec D_2$ if one of the following conditions holds.*

1. *D_1 is contained within the convex hull of D_2.*
2. *The convex hull of $D_1 \cup D_2$ contains two non-parallel line segments s and s' each connecting a point from D_1 with a point from D_2. Let x be the intersection point of the supporting lines of s and s'. Point x is closer to D_1 than to D_2 (in terms of the classic definition between a point and the set of points defined by the inside region of the quadrilateral).*

It is clear that we cannot have $D_1 \prec D_2$ and $D_2 \prec D_1$. It can, however, be that we have neither. As the drawings of A, B, and C correspond to simple quadrilaterals, we can show the following lemma (refer to Figure 3 for an illustration). Note that the statement makes sense, as A and B share the outer face.

▶ **Lemma 7.** *Let F be a frame containing an apex pair P_i^j and all four-cycles. We have that $\mathcal{D}(A) \prec \mathcal{D}(B)$ if $\mathcal{D}(A)$ is contained within an inner face of $\mathcal{D}(B + q_i^j)$. Equivalently, $\mathcal{D}(B) \prec \mathcal{D}(A)$ if $\mathcal{D}(B)$ is contained within an inner face of $\mathcal{D}(A + q_i^j)$.*

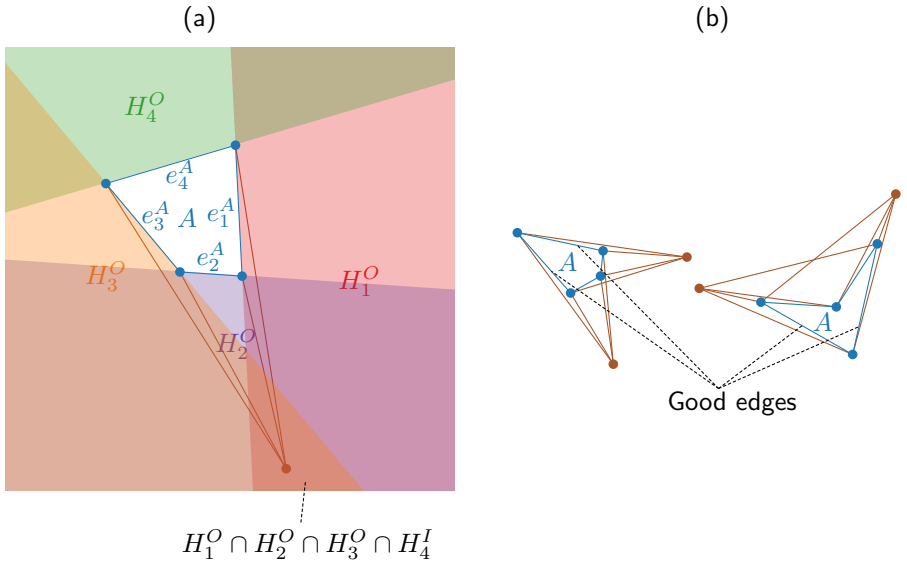


Figure 4 (a) The half-planes defined by $\mathcal{D}(A)$, an apex vertex must lie in a specific intersection of half-planes. (b) A convex and concave drawing of A with exactly two good edges.

Proof. Assume that $\mathcal{D}(A)$ is contained within an inner face of $\mathcal{D}(B + q_i^j)$. For the frame F, consider the drawing induced by q_i^j, A, and B. Consider the edges on the outer face of this drawing. Two of these connect a vertex of B with q_i^j; we denote these vertices as b and b'. Now, either A is contained within the convex hull of B, or the convex hull of $A + B$ contains two line segments s and s', each connecting a vertex of A with a vertex of B. In the former case, $\mathcal{D}(A) \prec \mathcal{D}(B)$ is immediate. In the latter case, s and s' are contained within the triangle defined by b, b', and q_i^j. Now notice that, if the intersection point of the lines defined by s and s' was closer to $\mathcal{D}(B)$ than to $\mathcal{D}(A)$, or if s and s' were parallel, the points on s and s' which are vertices of B would not be visible from q_i^j. Hence, we have $\mathcal{D}(A) \prec \mathcal{D}(B)$. ◀

As we cannot have both $\mathcal{D}(A) \prec \mathcal{D}(B)$ and $\mathcal{D}(B) \prec \mathcal{D}(A)$, we assume, w.l.o.g., that $\mathcal{D}(B) \not\prec \mathcal{D}(A)$. Thus, in any frame containing A, B, and some apex vertex q_i^j, B will lie on the outer face of the drawing defined by A and q_i^j.

We will now show the last ingredient, which can be seen as the key lemma of the construction. Essentially, we want to show that at most two edges of A are "good" – in the sense that they can lie on the outer face of the drawing of A plus some apex vertex (see Figure 4b). More formally, for a fixed drawing of A, the edge e_k^A, $k \in [4]$, is *good* if it can be the edge on the outer face of the planar straight-line drawing $\mathcal{D}(A + q_i^j)$ for some possible placement of some apex vertex q_i^j in the outer face of $\mathcal{D}(A)$. We show the following.

▶ **Lemma 8.** *For a fixed drawing $\mathcal{D}(A)$ of A, A has at most two good edges.*

Proof. The proof distinguishes two cases, depending on whether $\mathcal{D}(A)$ is convex or not. We start with the convex case. We show that it is not possible that e_1^A and e_3^A are both good. By symmetry, this means that e_2^A and e_4^A cannot be both good, and the statement follows. Thus, assume that e_1^A and e_3^A are good. We observe that in the drawing of A, each edge defines two open half-planes – the "outer half-plane" and the "inner half-plane" (see Figure 4a for the outer half-planes). More precisely, the outer half-plane defined by e_i^A contains as subset the points which can be connected by straight lines to a_i and $a_{i \pmod 4 + 1}$

without crossings, while the inner half plane does not. Thus, for $k \in [4]$, define H_k^O as the outer half-plane defined by e_k^A, and H_k^I as the inner half-plane. As the half-planes are open, they do not contain the line defined by their corresponding edge. Since e_1 is good, $H_1^I \cap H_2^O \cap H_3^O \cap H_4^O$ is non-empty (an apex vertex must be able to lie in this region for the drawing to be straight-line planar and for the edge e_1^A to be on the outer face). Similarly, as e_3 is good $H_1^O \cap H_2^O \cap H_3^I \cap H_4^O$ is non-empty. Note that $H_2^O \cap H_4^O$ is disjoint from the inner region defined by $\mathcal{D}(A)$. For H_1^O and H_1^I to both share points with the intersection of H_2^O and H_4^O, the line defined by the drawing of e_1^A must pass through $H_2^O \cap H_4^O$. Because the drawing of A is convex, and because $H_2^O \cap H_4^O$ does not share points with the region defined by $\mathcal{D}(A)$, this is not possible. We obtain a contradiction.

For the concave case, w.l.o.g., assume that the concave corner is at a_2, with its two incident edges e_1^A and e_2^A. We claim that none of e_1^A and e_2^A can be good. For e_1^A to be good, we again have that $H_1^I \cap H_2^O \cap H_3^O \cap H_4^O$ is non-empty. This is not possible: Consider any point x outside $\mathcal{D}(A)$ and in H_1^I. Notice that the line segment from x to a_2 intersects an edge of A. Hence, e_1^A cannot be good. The proof that e_2 cannot be good proceeds equivalently. ◀

Proof of Theorem 2. Assume for a contradiction that G admits a planar geometric storyplan. By our assumption, the drawings of A and B are in each other's outer face. Further $\mathcal{D}(B) \prec \mathcal{D}(A)$ does not hold. Thus, in each of the six frames existing by Corollary 4, $\mathcal{D}(B)$ is in the outer face of the drawing induced by A and the apex vertex. For the frame to be planar, the edge vertex of the apex-pair must be on the outer face of the drawing induced by the apex and A. Hence, the edge vertex must be connected with a good side of A. But, as there are only two good sides (Lemma 8), this is only possible for at most four apex-pairs. Hence, two of the edge vertices of the six apex pairs of Corollary 4 cannot be connected to a good side of A, a contradiction. ◀

4 NP-hardness

Furthermore, we can show that Problem 2 remains NP-hard.

▶ **Theorem 9.** *Deciding whether a graph admits a planar geometric storyplan is* NP*-hard.*

Essentially, we apply the same reduction from a variant of SAT as in [5]. Since every planar geometric storyplan is a planar storyplan, we get one direction of the correctness proof for free. However, we have to construct a geometric planar storyplan for every satisfying assignment. This requires a new construction, as the one in [5] does not use straight-line edges. For details, refer to the full version [13].

5 Conclusion

We have shown that not all graphs that admit a planar storyplan also admit a planar geometric storyplan, yet the decision problem remains NP-hard. A natural open question is whether the problem remains in NP, or whether it is complete for some other complexity class, such as $\exists\mathbb{R}$.

For further open problems on storyplans in general, we refer to those given in [5] and [14].

References

1 Fabian Beck, Michael Burch, Stephan Diehl, and Daniel Weiskopf. The state of the art in visualizing dynamic graphs. In Rita Borgo, Ross Maciejewski, and Ivan Viola, editors, *EuroVis 2014*. Eurographics Association, 2014. doi:10.2312/eurovisstar.20141174.

2 Carla Binucci, Ulrik Brandes, Giuseppe Di Battista, Walter Didimo, Marco Gaertler, Pietro Palladino, Maurizio Patrignani, Antonios Symvonis, and Katharina Anna Zweig. Drawing trees in a streaming model. In David Eppstein and Emden R. Gansner, editors, *Proc. Graph Drawing and Network Visualization (GD'09)*, volume 5849 of *Lecture Notes in Computer Science*, pages 292–303. Springer, 2009. doi:10.1007/978-3-642-11805-0_28.

3 Carla Binucci, Ulrik Brandes, Giuseppe Di Battista, Walter Didimo, Marco Gaertler, Pietro Palladino, Maurizio Patrignani, Antonios Symvonis, and Katharina Anna Zweig. Drawing trees in a streaming model. *Inf. Process. Lett.*, 112(11):418–422, 2012. doi:10.1016/j.ipl.2012.02.011.

4 Carla Binucci, Emilio Di Giacomo, William J. Lenhart, Giuseppe Liotta, Fabrizio Montecchiani, Martin Nöllenburg, and Antonios Symvonis. On the complexity of the storyplan problem. In Patrizio Angelini and Reinhard von Hanxleden, editors, *Proc. Graph Drawing and Network Visualization (GD'22)*, volume 13764 of *Lecture Notes in Computer Science*, pages 304–318. Springer, 2022. doi:10.1007/978-3-031-22203-0_22.

5 Carla Binucci, Emilio Di Giacomo, William J. Lenhart, Giuseppe Liotta, Fabrizio Montecchiani, Martin Nöllenburg, and Antonios Symvonis. On the complexity of the storyplan problem. *J. Comput. Syst. Sci.*, 139:103466, 2024. doi:10.1016/J.JCSS.2023.103466.

6 Manuel Borrazzo, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, and Maurizio Patrignani. Graph stories in small area. *J. Graph Algorithms Appl.*, 24(3):269–292, 2020. doi:10.7155/jgaa.00530.

7 Manuel Borrazzo, Giordano Da Lozzo, Fabrizio Frati, and Maurizio Patrignani. Graph stories in small area. In Daniel Archambault and Csaba D. Tóth, editors, *Proc. Graph Drawing and Network Visualization (GD'19)*, volume 11904 of *Lecture Notes in Computer Science*, pages 545–558. Springer, 2019. doi:10.1007/978-3-030-35802-0_41.

8 Giordano Da Lozzo and Ignaz Rutter. Planarity of streamed graphs. *Theor. Comput. Sci.*, 799:1–21, 2019. doi:10.1016/J.TCS.2019.09.029.

9 Giuseppe Di Battista, Walter Didimo, Luca Grilli, Fabrizio Grosso, Giacomo Ortali, Maurizio Patrignani, and Alessandra Tappini. Small point-sets supporting graph stories. In Patrizio Angelini and Reinhard von Hanxleden, editors, *Proc. Graph Drawing and Network Visualization (GD'22)*, volume 13764 of *Lecture Notes in Computer Science*, pages 289–303. Springer, 2022. doi:10.1007/978-3-031-22203-0_21.

10 Giuseppe Di Battista, Walter Didimo, Luca Grilli, Fabrizio Grosso, Giacomo Ortali, Maurizio Patrignani, and Alessandra Tappini. Small point-sets supporting graph stories. *J. Graph Algorithms Appl.*, 27(8):651–677, 2023. doi:10.7155/JGAA.00639.

11 Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, Fabrizio Montecchiani, and Ioannis G. Tollis. Exploring complex drawings via edge stratification. In Stephen K. Wismath and Alexander Wolff, editors, *Proc. Graph Drawing and Network Visualization (GD'13)*, volume 8242 of *Lecture Notes in Computer Science*, pages 304–315. Springer, 2013. doi:10.1007/978-3-319-03841-4_27.

12 Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, Fabrizio Montecchiani, and Ioannis G. Tollis. Techniques for edge stratification of complex graph drawings. *J. Vis. Lang. Comput.*, 25(4):533–543, 2014. doi:10.1016/j.jvlc.2014.05.001.

13 Alexander Dobler, Maximilian Holzmüller, and Martin Nöllenburg. Geometry matters in planar storyplans. *CoRR*, abs/2508.12747, 2025. arXiv:2508.12747.

14 Jirí Fiala, Oksana Firman, Giuseppe Liotta, Alexander Wolff, and Johannes Zink. Outerplanar and forest storyplans. In Henning Fernau, Serge Gaspers, and Ralf Klasing, editors, *Proc. Theory and Practice of Computer Science (SOFSEM'24)*, volume 14519 of *Lecture Notes in Computer Science*, pages 211–225. Springer, 2024. doi:10.1007/978-3-031-52113-3_15.

15 Michael T. Goodrich and Pawel Pszona. Streamed graph drawing and the file maintenance problem. In Stephen K. Wismath and Alexander Wolff, editors, *Proc. Graph Drawing and Network Visualization (GD'13)*, volume 8242 of *Lecture Notes in Computer Science*, pages 256–267. Springer, 2013. doi:10.1007/978-3-319-03841-4_23.

16 Maximilian Holzmüller. Expanding the planar storyplan problem. Master's thesis, Technische Universität Wien, 2025. URL: https://repositum.tuwien.at/handle/20.500.12708/213348.

17 Corinna Vehlow, Fabian Beck, and Daniel Weiskopf. Visualizing dynamic hierarchies in graph sequences. *IEEE Trans. Vis. Comput. Graph.*, 22(10):2343–2357, 2016. doi:10.1109/TVCG.2015.2507595.

Treewidth of Outer k-Planar Graphs

Rafał Pyzik
Institute of Theoretical Computer Science, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland

Abstract

Treewidth is an important structural graph parameter that quantifies how closely a graph resembles a tree-like structure. It has applications in many algorithmic and combinatorial problems. In this paper, we study the treewidth of *outer k-planar* graphs, that is, graphs admitting a *convex drawing* (a straight-line drawing where all vertices lie on a circle) in which every edge crosses at most k other edges. We also consider the more general class of *outer min-k-planar* graphs, which are graphs admitting a convex drawing where for every crossing of two edges at least one of these edges is crossed at most k times.

Firman, Gutowski, Kryven, Okada and Wolff [GD 2024] proved that every outer k-planar graph has treewidth at most $1.5k + 2$ and provided a lower bound of $k + 2$ for even k. We establish a lower bound of $1.5k + 0.5$ for every odd k. Additionally, they showed that every outer min-k-planar graph has treewidth at most $3k + 1$. We improve this upper bound to $3 \cdot \lfloor k/2 \rfloor + 4$.

Our approach also allows us to upper bound the *separation number*, a parameter closely related to treewidth, of outer min-k-planar graphs by $2 \cdot \lfloor k/2 \rfloor + 4$. This improves upon the previous bound of $2k + 1$ and achieves a bound with an optimal multiplicative constant.

2012 ACM Subject Classification Mathematics of computing → Graph theory

Keywords and phrases treewidth, outer k-planar graphs, outer min-k-planar graphs, separation number

Digital Object Identifier 10.4230/LIPIcs.GD.2025.28

Related Version *Full Version*: `https://arxiv.org/abs/2506.08151`

Acknowledgements I thank Grzegorz Gutowski for introducing me to this work, for many helpful discussions, and for proofreading this paper. I am also grateful to the reviewers for their insightful comments and suggestions.

1 Introduction

In this paper, we study classes of graphs admitting a *convex drawing* with a bounded number of edge crossings. A convex drawing is a straight-line drawing with all vertices drawn on a common circle. Bannister and Eppstein [1, 2] proved that the treewidth of graphs admitting a convex drawing with at most k crossings in total is bounded by a linear function of $\sqrt{k}$. For fixed k, they also provided a linear-time algorithm deciding whether a given graph admits such a drawing (using Courcelle's theorem [5]). Another well-studied class of graphs in this context is the class of *outer k-planar* graphs, that is, graphs that admit a convex drawing in which every edge crosses at most k other edges. These graphs have treewidth bounded by a linear function of k, which was first proven by Wood and Telle [13, Proposition 8.5]. The authors of [4], also using Courcelle's theorem, presented, for any fixed k, a linear-time algorithm that tests whether a given graph is maximal outer k-planar. Recently, Kobayashi, Okada and Wolff [10], for any fixed k, provided a polynomial-time algorithm to test whether a given graph is outer k-planar and proved that recognising outer k-planar graphs is XNLP-hard.


© Rafał Pyzik;
licensed under Creative Commons License CC-BY 4.0

For disambiguation, we recall the definition of k-*outerplanar* graphs. A graph is *outerplanar* if it has a planar drawing with all vertices lying on the outer face. A graph is 1-*outerplanar* when it is outerplanar. A graph is k-*outerplanar* for $k > 1$ when it has a planar drawing such that after removing the vertices of the outer face, each of the remaining components is $(k-1)$-outerplanar.

We mainly study the treewidth of outer k-planar graphs and *outer min-k-planar* graphs. A graph is outer min-k-planar if it admits a convex drawing in which, for every crossing of two edges, at least one of these edges is crossed at most k times. Firman, Gutowski, Kryven, Okada and Wolff [7] proved that outer k-planar graphs have treewidth at most $1.5k + 2$ and outer min-k-planar graphs have treewidth at most $3k + 1$. To obtain these results, they showed that every outer k-planar graph admits a *triangulation* of the outer cycle such that every edge of the triangulation is crossed at most k times by the edges of the graph. A similar property was proven for outer min-k-planar graphs.

Another property closely related to treewidth is the separation number of a graph. A *separation* of a graph G is a pair (A, B) of subsets of $V(G)$ such that $A \cup B = V(G)$ and there are no edges between the sets $A \setminus B$ and $B \setminus A$. The *order* of a separation is $|A \cap B|$. A separation is *balanced* if $|A \setminus B| \leq \frac{2}{3}|V(G)|$ and $|B \setminus A| \leq \frac{2}{3}|V(G)|$. The *separation number* of a graph G, denoted $\mathrm{sn}(G)$, is the minimum integer a such that every subgraph of G has a balanced separation of order at most a. Robertson and Seymour [11] proved that $\mathrm{sn}(G) \leq \mathrm{tw}(G) + 1$ for every graph G. From the other side, Dvořák and Norin [6] showed that $\mathrm{tw}(G) \leq 15\,\mathrm{sn}(G)$. Recently, Houdrouge, Miraftab and Morin [8] provided a more constructive proof of an analogous inequality, but with a worse multiplicative constant.

Our contribution. The authors of [7] proved that every outer k-planar graph has treewidth at most $1.5k + 2$. They also presented a lower bound of $k + 2$ for every even k. We present an infinite family of outer k-planar graphs with treewidth at least $1.5k + 0.5$, showing that the multiplicative constant 1.5 in the upper bound cannot be improved; see Section 3.

We also improve the upper bounds for the treewidth and separation number of outer min-k-planar graphs. It was previously known that the treewidth of such graphs is at most $3k+1$ and the separation number is at most $2k+1$ [7]. We give an upper bound of $3 \cdot \lfloor k/2 \rfloor + 4$ for the treewidth (see Section 4) and an upper bound of $2 \cdot \lfloor k/2 \rfloor + 4$ for the separation number (see Section 5). Both multiplicative constants are optimal, as the lower bounds for outer k-planar graphs also hold for outer min-k-planar graphs – namely, our lower bound of $1.5k + 0.5$ for the treewidth and the lower bound of $k + 2$ for the separation number presented in [7].

2 Preliminaries

Let G be a graph. By $V(G)$ and $E(G)$ we denote the set of vertices and edges of G, respectively. For an edge that connects vertices u and v, we use the compact notation uv, instead of $\{u, v\}$. For a directed edge, we use the standard notation (u, v). Let $\deg(v)$ denote the degree of a vertex v, and let $\Delta(G)$ denote the maximum degree of a vertex of G.

For a graph G, a subgraph *induced* by a set $U \subseteq V(G)$, denoted $G[U]$, is a subgraph with vertex set U and all edges of G between the vertices of U. A *spanning tree* of a graph G is a subgraph of G containing all the vertices of G that is a tree. By $\mathrm{dist}_G(v, w)$, we denote the distance (i.e. the length of the shortest path) between v and w in a graph G. For any tree T rooted at vertex r, we define the depth of a vertex v as $\mathrm{depth}_T(v) = \mathrm{dist}_T(r, v)$. We may omit subscripts if they are clear from the context.

A *tree decomposition* $\mathcal{T} = (T, B)$ of a graph G is a collection of *bags*, $\{B_x : x \in V(T)\}$, indexed by the vertices of a tree T. The bags are subsets of $V(G)$ and satisfy the following properties:

1. for every vertex $v \in V(G)$, the set $\{x : v \in B_x\}$ induces a non-empty subtree of T;
2. for every edge $uv \in E(G)$, there exists a bag containing both u and v.

The *width* of a given tree decomposition is the size of the largest bag minus one. The *treewidth* of a graph G, denoted by $\mathrm{tw}(G)$, is the minimum width of any tree decomposition of G.

A set $\mathcal{B}$ of non-empty subsets of $V(G)$ is a *bramble* if:

1. for every $X \in \mathcal{B}$, the induced subgraph $G[X]$ is connected;
2. for every $X_1, X_2 \in \mathcal{B}$, the induced subgraph $G[X_1 \cup X_2]$ is connected. In other words, each $X_1, X_2 \in \mathcal{B}$ either share a common vertex or there exists an edge of G incident to both X_1 and X_2.

A *hitting set* of a bramble is a set of vertices with non-empty intersection with every element of $\mathcal{B}$. The *order* of a bramble is the size of its smallest hitting set. The *bramble number* of a graph G, denoted by $\mathrm{bn}(G)$, is the maximum order of any bramble of G.

The following result by Seymour and Thomas shows the relation between the bramble number and treewidth.

▶ **Theorem 1** (Seymour and Thomas, [12]). *For every graph G, $\mathrm{tw}(G) = \mathrm{bn}(G) - 1$.*

We say that a graph G is a *minor* of a graph H if G can be obtained from H by a sequence of vertex deletions, edge deletions or edge contractions. The contraction of an edge uv is an operation that replaces vertices u and v with a new vertex adjacent to every vertex other than u and v that was adjacent to u or v. It is a well-known fact that if G is a minor of H, then $\mathrm{tw}(G) \leq \mathrm{tw}(H)$. A proof of this fact can be found in [3].

In the remainder of this section, we introduce some notation and simple observations regarding drawings. A *convex drawing* of a graph G is a straight-line drawing where the vertices of G are placed on distinct points of a circle. Given a cyclic order $(v_1, \ldots, v_n)$ of vertices, we say that an edge v_iv_j with $i < j$ *crosses* an edge $v_{i'}v_{j'}$ with $i' < j'$ if either $1 \leq i < i' < j < j' \leq n$ or $1 \leq i' < i < j' < j \leq n$. We only consider convex drawings where no three pairwise non-adjacent edges pass through the same point. An *outer k-planar drawing* of a graph is a convex drawing such that every edge crosses at most k other edges. An *outer min-k-planar drawing* of a graph is a convex drawing such that for every crossing of two edges, at least one of these edges crosses at most k other edges.

An outer min-k-planar graph G is *maximal outer min-k-planar* if for every $\{u, v\} \subseteq V(G)$ with $uv \notin E(G)$, the graph $G + uv$ is not outer min-k-planar.

▶ **Observation 2.** *Let G be a maximal outer min-k-planar graph with at least three vertices. Then, in every outer min-k-planar drawing of G, the outer face is bounded by a cycle.*

Proof. Consider an outer min-k-planar drawing Γ of G. Let u and v be consecutive vertices in the cyclic order defined by Γ. Suppose, for contradiction, that $uv \notin E(G)$. Note that the graph $G + uv$ has an outer min-k-planar drawing defined by the same cyclic order as Γ, contradicting the maximality of G. ◀

A graph G is *expanded outer min-k-planar* if G is an outer min-k-planar graph with $\Delta(G) \leq 3$ and its outer face is bounded by a cycle in some outer min-k-planar drawing of G.

▶ **Observation 3.** *Every outer min-k-planar graph G is a minor of an expanded outer min-k-planar graph G'.*

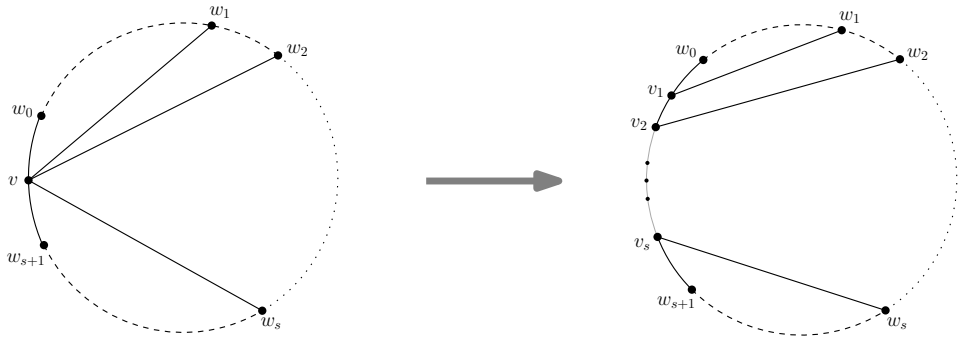


Figure 1 The transformation described in Observation 3.

Proof. Let us assume that G is maximal outer min-k-planar. Now, in order to obtain G' from G, we perform the following transformation to every vertex v of G with $\deg(v) \geq 4$. The transformation is depicted in Figure 1. Let $w_0, w_1, \ldots, w_s, w_{s+1}$ be all neighbors of v in clockwise order, with edges vw_0 and vw_{s+1} incident to the outer face of G. We replace v with a path $v_1, \ldots, v_s$, put it on the outer face of G in counterclockwise order, in the place of v. We connect this path to vertices w_0 and w_{s+1} by adding edges v_1w_0 and v_sw_{s+1}. Finally, for every $1 \leq i \leq s$, we add an edge v_iw_i that corresponds to an edge vw_i in the original graph. It is easy to see that G is a minor of G' and the ordering of corresponding edges in G' matches the one in G. Moreover, the crossings in the resulting graph naturally correspond to the crossings in the original graph. ◀

The vertices $v_1, \ldots, v_s$ obtained in the proof as a replacement of v are called *images* of v. The vertex v is the *origin* of these vertices, which we denote as $\mathrm{org}(v_i) = v$. If the transformation was not performed for some vertex v of G, i.e. $\deg(v) \leq 3$, then v is an image and origin of itself.

Since removing edges increases neither the treewidth nor the separation number, we are interested in the properties of maximal outer min-k-planar graphs. Also, taking a minor does not increase treewidth, so we work with expanded graphs when establishing upper bounds on treewidth.

3 Lower bound on the treewidth of outer k-planar graphs

In this section, we construct an infinite family of outer k-planar graphs with treewidth at least $1.5k + 0.5$. This improves the previous lower bound of $k + 2$ that was presented in [7]. We begin by defining the necessary graphs.

For positive integers m and n, let $X_{m,n}$ denote the grid of m rows and n columns, i.e. a graph with

$$V(X_{m,n}) = \{x_{i,j} : 1 \leq i \leq m, 1 \leq j \leq n\} \text{ and } E(X_{m,n}) = \{x_{i,j}x_{k,l} : |i-k| + |j-l| = 1\}.$$

For a positive integer k, let Q_k be a copy of the grid $X_{2k,2k}$ and let R_k be a copy of $X_{2k(k+1),k}$. Denote by $v_{i,j}$, for $1 \leq i, j \leq 2k$, the vertex in the i-th row and j-th column of Q_k, and by $u_{i,j}$, for $1 \leq i \leq 2k(k+1)$, $1 \leq j \leq k$, the vertex in the i-th row and j-th column of R_k. Let G_k be a graph such that $V(G_k) = V(Q_k) \cup V(R_k)$ and

$$E(G_k) = E(Q_k) \cup E(R_k) \cup \left\{v_{i,2k}u_{(i-1)(k+1)+j,1} : 1 \leq i \leq 2k, 1 \leq j \leq k+1\right\};$$

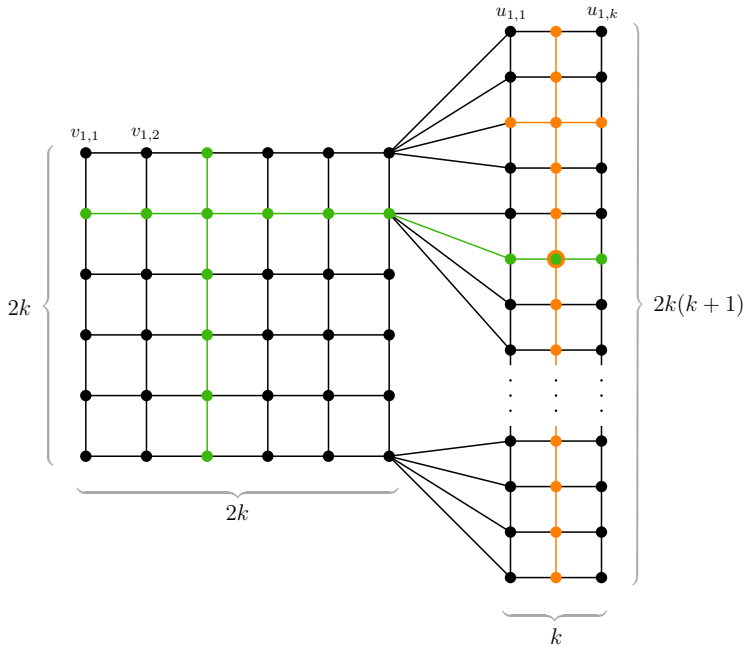


Figure 2 The graph G_k, for $k = 3$, with a subgraph of $\mathcal{B}_1$ colored green and a subgraph of $\mathcal{B}_2$ colored orange.

see Figure 2. For $1 \leq i \leq 2k(k+1)$, let the i-th *extended row* of G_k be the union of the i-th row of R_k and the $\lceil \frac{i}{k+1} \rceil$-th row of Q_k. Note that each row of Q_k is contained in $k+1$ extended rows and the graph induced by each extended row is a path.

The graph G_k was previously introduced by Kammer and Tholey [9] as an example of tightness of the upper bound on the treewidth of k-outerplanar graphs. They used the *cops and robber game* to establish a lower bound on the treewidth of G_k. Below, we present a proof using brambles.

▶ **Theorem 4** (Kammer and Tholey, [9]). *For every $k \geq 1$, $\mathrm{tw}(G_k) = 3k - 1$.*

Proof. Notice that the drawing of G_k in Figure 2 is k-outerplanar. By the fact that k-outerplanar graphs have treewidth at most $3k-1$ [3, Theorem 83], we get $\mathrm{tw}(G_k) \leq 3k-1$.

To prove that $\mathrm{tw}(G_k) \geq 3k-1$, we construct a bramble of order $3k$. Then, using Theorem 1, we get $\mathrm{tw}(G_k) \geq 3k-1$. Let $\mathcal{B}_1$ be a family consisting of every subset of $V(G_k)$ that is a union of an extended row of G_k and a column of Q_k. Let $\mathcal{B}_2$ be a family consisting of every subset of $V(G_k)$ that is a union of a row of R_k and a column of R_k. The set $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$ forms a bramble of G_k, since each subgraph induced by an element of $\mathcal{B}$ is connected and every two such subgraphs have at least one common vertex.

Consider any hitting set S of $\mathcal{B}$. Let q and r be the number of vertices of S in $V(Q_k)$ and in $V(R_k)$, respectively. We would like to show that $|S| = q + r \geq 3k$. Note that $r \geq k$, as otherwise there is a row and a column of R_k not containing any element of S, and thus there is an element of $\mathcal{B}_2$ not hit by S.

If $q \geq 2k$, then $q + r \geq 3k$. Otherwise, let $q = 2k - l$ for some positive integer l. Now, we can find at least l columns and at least l rows of Q_k not intersecting S. These l rows are contained in $l(k+1)$ extended rows. Each of them has to intersect S at some vertex of R_k, because otherwise we can find a column of Q_k and an extended row not intersecting S that

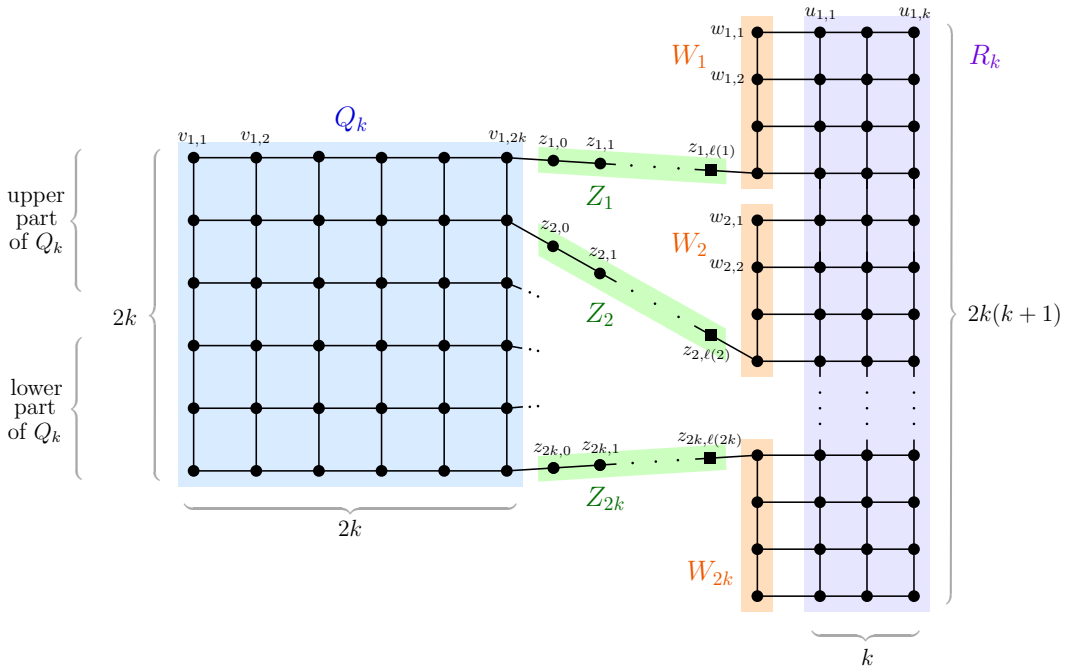


Figure 3 The graph F_k, which is a modification of the graph G_k, for $k = 3$.

form an element of $\mathcal{B}_1$. The extended rows restricted to R_k are pairwise disjoint, so we have $r \geq l(k+1)$. Summing up, we get $q + r \geq 2k - l + l(k+1) = 2k + lk \geq 2k + k = 3k$, which concludes the proof. ◀

Let F_k be the following modification of G_k depicted in Figure 3. We set $\ell(i) = (k-i)(k+1)$ for $1 \leq i \leq k$, and $\ell(i) = (i-k-1)(k+1)$ for $k+1 \leq i \leq 2k$. We remove every edge between the grids Q_k and R_k. For every $1 \leq i \leq 2k$, we add a path Z_i of length $\ell(i)$ on new vertices $z_{i,0}, z_{i,1}, \ldots, z_{i,\ell(i)}$. Note that

$$k^2 - 1 = |V(Z_1)| = |V(Z_{2k})| > |V(Z_2)| = |V(Z_{2k-1})| > \ldots > |V(Z_k)| = |V(Z_{k+1})| = 1.$$

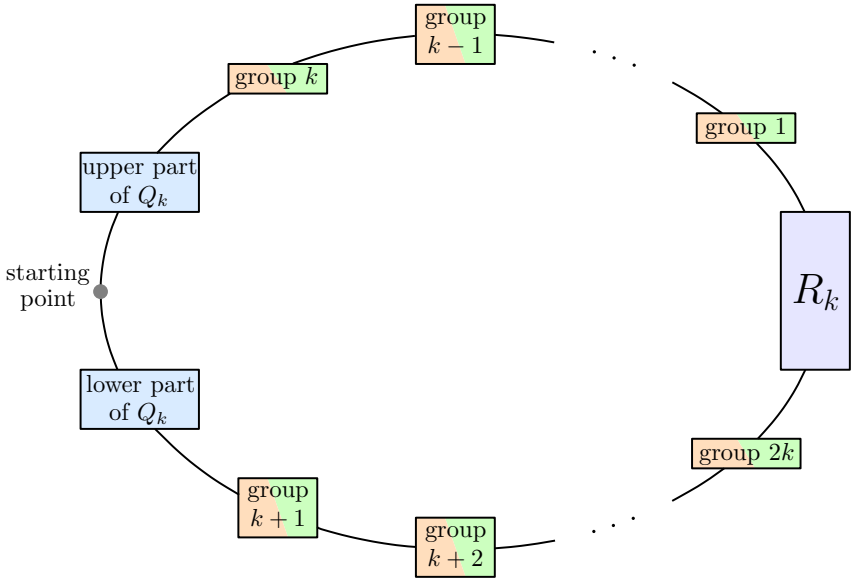
We also add a path W_i of length k on new vertices $w_{i,1}, w_{i,2}, \ldots, w_{i,k+1}$. We connect $v_{i,2k}$ with Z_i by adding the edge $v_{i,2k}z_{i,0}$. Next, we connect Z_i with W_i by adding the edge $z_{i,\ell(i)}w_{i,k+1}$ for $1 \leq i \leq k$, or the edge $z_{i,\ell(i)}w_{i,1}$ for $k+1 \leq i \leq 2k$. Finally, we connect W_i with R_k by adding the edges $w_{i,j}u_{(i-1)(k+1)+j,1}$ for every $1 \leq j \leq k+1$.

To see that G_k is a minor of F_k, it is enough to contract, for every $1 \leq i \leq 2k$, vertex $v_{i,2k}$ with all vertices of the paths Z_i and W_i. Since taking a minor does not increase the treewidth, we obtain the following corollary.

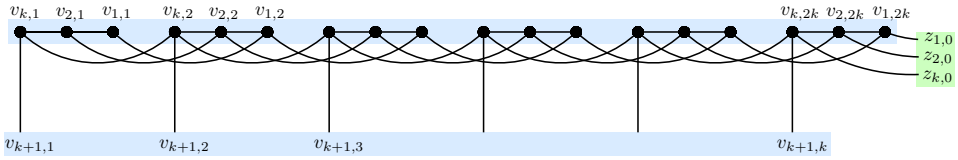
▶ **Corollary 5.** *For every $k \geq 1$, $\mathrm{tw}(F_k) \geq 3k - 1$.*

▶ **Theorem 6.** *For every $k \geq 1$, The graph F_k has an outer $(2k-1)$-planar drawing.*

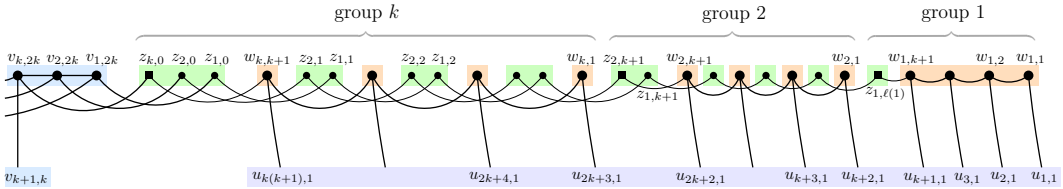
Proof. We describe an outer $(2k-1)$-planar drawing of F_k, as depicted in Figure 4. We call the set of vertices $\{v_{i,j} : 1 \leq i \leq k, 1 \leq j \leq 2k\}$ the *upper part* of Q_k. The other vertices of Q_k are called the *lower part* of Q_k. We define a cyclic order of the vertices of F_k by arranging them in clockwise order from some selected starting point on a circle.



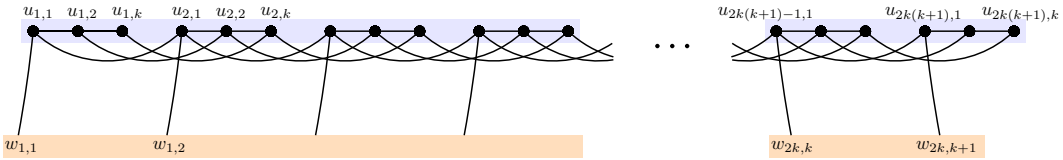
(a) Overview of the outer $(2k-1)$-planar drawing of F_k.



(b) Drawing of the upper part of the grid Q_k.



(c) Drawing of the groups $k, k-1, \ldots, 1$ connected to the upper part of Q_k.



(d) Drawing of the grid R_k.

Figure 4 Fragments of the outer $(2k-1)$-planar drawing of F_k, where $k=3$. The drawings are not shown as straight-line drawings, but illustrate the order in which the vertices are placed. A straight-line drawing can be easily obtained by placing vertices in this order on the circle.

First, we place the vertices of the upper part of Q_k in column-by-column order (see Figure 4b):

$$v_{k,1}, \ldots, v_{2,1}, v_{1,1}, \quad v_{k,2}, \ldots, v_{2,2}, v_{1,2}, \quad \ldots \quad v_{k,2k}, \ldots, v_{2,2k}, v_{1,2k}.$$

We divide the vertices of paths Z_i and W_i, for $1 \leq i \leq k$, into k groups, as follows. The i-th group contains the vertices of W_i and the vertex $z_{i,\ell(i)}$. If $i \geq 2$, the i-th group also contains vertices $z_{a,b}$ for $1 \leq a < i$ and $(k-i)(k+1) \leq b < (k-i+1)(k+1)$. In the drawing, we place the groups of indices $k, k-1, \ldots, 1$, in this order. We arrange the vertices in the i-th group, for $2 \leq i \leq k$, in the order (see Figure 4c):

$$z_{i,\ell(i)},$$
$$z_{i-1,(k-i)(k+1)}, z_{i-2,(k-i)(k+1)}, \cdots, z_{1,(k-i)(k+1)}, w_{i,k+1},$$
$$z_{i-1,(k-i)(k+1)+1}, z_{i-2,(k-i)(k+1)+1}, \cdots, z_{1,(k-i)(k+1)+1}, w_{i,k},$$
$$\vdots$$
$$z_{i-1,(k-i+1)(k+1)-1}, z_{i-2,(k-i+1)(k+1)-1}, \cdots, z_{1,(k-i+1)(k+1)-1}, w_{i,1}.$$

The group of index 1 has vertices arranged in the order: $z_{1,\ell(1)}, w_{1,k+1}, w_{1,k}, \ldots, w_{1,1}$.

Next, we put the vertices of R_k in row-by-row order (see Figure 4d):

$$u_{1,1}, u_{1,2}, \ldots, u_{1,k}, \quad u_{2,1}, u_{2,2}, \ldots, u_{2,k}, \quad \ldots \quad u_{2k(k+1),1}, u_{2k(k+1),2}, \ldots, u_{2k(k+1),k}.$$

The vertices of F_k that are not placed yet are in the lower part of Q_k or in the paths Z_i, W_i with $k+1 \leq i \leq 2k$. We arrange them in counterclockwise order from the starting point and place them between the starting point and the vertices of R_k. The order is symmetric, with respect to the starting point, to the one used to arrange the upper part of Q_k and the paths Z_i, W_i with $1 \leq i \leq k$. Every vertex $v_{i,j}$, where $k+1 \leq i \leq 2k$ and $1 \leq j \leq 2k$, is placed symmetrically to $v_{2k-i+1,j}$. Vertex $z_{i,j}$, where $k+1 \leq i \leq 2k$ and $0 \leq j \leq \ell(i)$, is placed symmetrically to $z_{2k-i+1,j}$, and vertex $w_{i,j}$, where $k+1 \leq i \leq 2k$ and $1 \leq j \leq k+1$, is placed symmetrically to $w_{2k-i+1,k-j+2}$. The symmetrical drawing of the i-th group, for every $1 \leq i \leq k$, forms the group of index $2k-i+1$.

Now, we partition the edges into several numbered types. For the edges of each type, we show that they cross at most $2k-1$ other edges.

Edges of Q_k:

1. The "column" edges of the upper or lower part of Q_k, that is, the edges $v_{i,j}v_{i+1,j}$, for $1 \leq i \leq 2k-1, i \neq k$ and $1 \leq j \leq 2k$. They cross no other edges.
2. The "column" edges between the upper part and the lower part of Q_k, that is, the edges $v_{k,j}v_{k+1,j}$, for $1 \leq j \leq 2k$. Each of these edges crosses $k-1$ edges of type 3 of the upper part of Q_k, and $k-1$ edges of type 3 of the lower part of Q_k. The edge $v_{k,1}v_{k+1,1}$ does not cross any edges.
3. The "row" edges of Q_k, that is, the edges $v_{i,j}v_{i,j+1}$, for $1 \leq i \leq 2k$ and $1 \leq j \leq 2k-1$. Each of these edges crosses at most $2(k-1)$ edges of type 3 (and type 4, for $j = 2k-1$), and additionally at most one edge of type 2.

Edges between Q_k and the groups:

4. Each edge $v_{i,2k}z_{i,0}$, for $1 \leq i \leq k$, crosses $2(k-1)$ edges in total: $2(i-1)$ edges of types 3, 4 incident to vertices $v_{j,2k}$, for all $j \in \{1, \ldots, i-1\}$; and $2(k-i)$ edges incident to vertices $z_{j,0}$, for all $j \in \{i+1, \ldots, k\}$. By symmetry, each edge $v_{i,2k}z_{i,0}$, for $k+1 \leq i \leq 2k$, also crosses exactly $2(k-1)$ edges.

Edges of the groups:

5. Each edge $z_{i,\ell(i)}w_{i,k+1}$, for $1 \leq i \leq k$, crosses exactly $2(i-1)$ edges incident to the vertices $z_{i-1,\ell(i)}, \ldots, z_{1,\ell(i)}$. By symmetry, each edge $z_{i,\ell(i)}w_{i,1}$, for $k+1 \leq i \leq 2k$, also crosses exactly $2(2k-i)$ edges.
6. Each edge of the path Z_i, for $1 \leq i \leq 2k$, crosses at most $2(k-2)$ edges incident to at most $k-2$ vertices of some paths Z_j with $j \in \{1, \ldots, 2k\}$, and at most three edges incident to some vertex $w_{a,b} \in V(W_j)$ with $j \in \{1, \ldots, 2k\}$.
7. Each edge of the path W_i, for $1 \leq i \leq k$, crosses $2(i-1)$ edges incident to $i-1$ vertices of some paths Z_j with $j \in \{1, \ldots, k\}$. By symmetry, each edge of the path W_i, for $k+1 \leq i \leq 2k$, also crosses $2(2k-i)$ edges.

Edges between the groups and R_k:

8. Each edge $w_{i,j}u_{(i-1)(k+1)+j,1}$, for $1 \leq i \leq 2k$ and $1 \leq j \leq k+1$, crosses at most $k-1$ edges of some paths Z_a with $a \in \{1, \ldots, 2k\}$, and at most $k-1$ edges of R_k of type 10.

Edges of R_k:

9. The "row" edges of R_k, that is, the edges $u_{i,j}u_{i,j+1}$, for $1 \leq i \leq 2k(k+1)$ and $1 \leq j \leq k-1$. They cross no other edges.
10. The "column" edges of R_k, that is, the edges $u_{i,j}u_{i+1,j}$, for $1 \leq i \leq 2k(k+1)-1$ and $1 \leq j \leq k$. Each of these edges crosses at most $2(k-1)$ other edges of this type and at most one edge of type 8. ◀

▶ **Theorem 7.** *For every odd positive integer k, there exists an outer k-planar graph G with* $\mathrm{tw}(G) \geq 1.5k + 0.5$.

Proof. By Theorem 6, the graph $F_{\frac{k+1}{2}}$ is outer k-planar, and by Corollary 5, it has treewidth at least $3 \cdot \frac{k+1}{2} - 1 = 1.5k + 0.5$. ◀

4 Upper bound on the treewidth of outer min-k-planar graphs

In this section, we establish an upper bound on the treewidth of outer min-k-planar graphs. We improve the previous bound of $3k+1$ presented in [7] to $3 \cdot \lfloor k/2 \rfloor + 4$. We begin by introducing required notation.

For an outer min-k-planar graph G with a given drawing Γ, we define the *planarization* G_P (with respect to Γ) as the graph whose vertex set is the union of $V(G)$ and all crossing points of edges of G. We say that a vertex $w \in V(G_P)$ *lies* on an edge $uv \in E(G)$ if w is an endpoint of uv, or if the crossing point corresponding to w belongs to the segment representing the drawing of uv in Γ. Graph G_P contains an edge between two vertices if and only if they are consecutive vertices lying on the drawing of some edge of G. Observe that G_P is a planar graph. We say that an edge $xy \in E(G_P)$ *lies* on an edge $uv \in E(G)$ if both x and y lie on uv in Γ. Furthermore, we say that a vertex $v \in V(G_P)$ is *outer* if it is incident to the outer face of G_P. Otherwise, v is an *inner* vertex. As we consider only maximal outer min-k-planar graphs G, the outer vertices of G_P are exactly the vertices of G, while the inner vertices of G_P are exactly the crossing points of edges of G. By f_o we denote the outer face of G_P.

For a planar graph G, let G^* denote its dual graph. Let $f^* \in V(G^*)$ denote the vertex dual to the face f of G, and let $e^* \in E(G^*)$ denote the edge dual to the edge $e \in E(G)$. We remark that G^* can be drawn on the drawing of G in a way that f^* is on the face f and the drawing of e^* is a curve that crosses the edge e exactly once and passes only through the faces corresponding to the endpoints of e^*.

The following lemma shows a bijection between a spanning tree T of a planar graph G and a spanning tree T^* of G^*, where $T^* = \text{dual}(T)$. We also use the notation $\text{dual}(T^*)$ to denote T.

▶ **Lemma 8** (Folklore). *Let T be a spanning tree of a planar graph G. Then T^* with $V(T^*) = V(G^*)$ and $E(T^*) = \{e^* : e \in E(G) \setminus E(T)\}$ is a spanning tree of G^*.*

The next lemma proves that there exists a spanning tree preserving shortest paths from a given vertex. Such a tree can be found using a breadth-first search.

▶ **Lemma 9** (Folklore). *Let G be a graph and let r be a vertex of G. Then there exists a spanning tree T of G rooted at r such that $\text{depth}_T(v) = \text{dist}_G(r, v)$ for every vertex v of G.*

▶ **Lemma 10.** *Let G be an expanded outer min-k-planar graph with its planarization G_P. Then $\text{dist}(f^*, f_o^*) \leq \lfloor k/2 \rfloor + 1$ for every vertex $f^* \in V(G_P^*)$.*

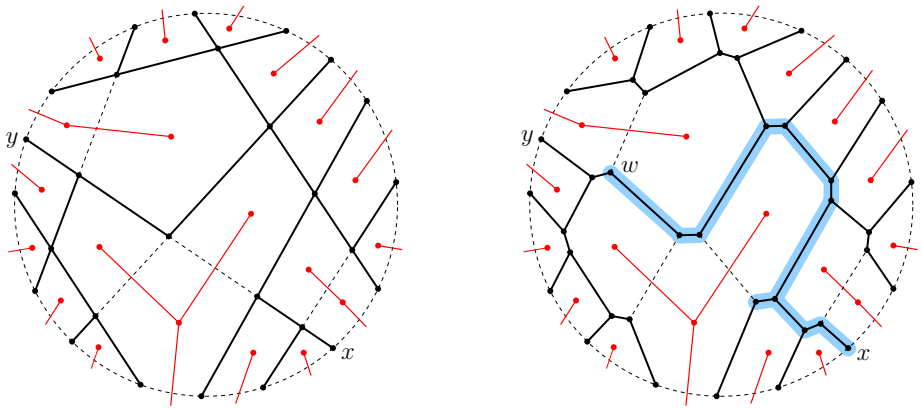
Proof. Let f be an inner face of G_P. If f is adjacent to f_o, then $\text{dist}(f^*, f_o^*) = 1$. Otherwise, let v be a vertex of G_P incident to f. Since G is expanded, the vertex v is inner, and hence it lies on an edge e of G that crosses at most k other edges. Let $v_0, v_1, \ldots, v_s, v_{s+1}, \ldots, v_{s+t+1}$ be all vertices lying on e, listed in the order along e, where $v_s = v$ and v_{s+1} is a neighbor of v that is incident to f. We may assume that $s \leq t$, i.e. v_s is closer to an endpoint of the edge e than v_{s+1} to the other endpoint of e. Note that at most $k + 2$ vertices lie on e (two endpoints and at most k crossing points), so $s + t + 2 \leq k + 2$. Together with the previous inequality, this implies $s \leq k/2$. The number s is an integer so $s \leq \lfloor k/2 \rfloor$.

We inductively define a sequence $w_s, w_{s-1}, \ldots, w_0$ of vertices. Vertex w_s is the neighbor of v_s that is incident to f and does not lie on e. For every $i = s - 1, \ldots, 0$, the vertex w_i is one of the two neighbors of v_i not lying on e, chosen so that w_i, v_i, v_{i-1} and w_{i-1} are incident to the same face of G_P. Let e_i denote the edge $v_i w_i$. Note that the path formed by the edges $e_s^*, \ldots, e_0^*$ connects f^* with f_o^* (since e_0 is incident to f_o). Hence, $\text{dist}(f^*, f_o^*) \leq s + 1 \leq \lfloor k/2 \rfloor + 1$. ◀

Let v be an inner vertex of G_P. Since we forbid common crossing points of three edges of G, the vertex v has four neighbors, which we denote by w_1, w_2, w_3, w_4 in clockwise order. The *splitting* of the vertex v replaces it with two vertices v_1 and v_2 connected by an edge, and adds edges $v_1 w_1$, $v_1 w_2$ and $v_2 w_3$, $v_2 w_4$. We fix a planar embedding of the new graph by placing v_1, v_2 very close to where v was drawn. We say that vertices v_1 and v_2 *lie* on the same edges as the vertex v. Moreover, we call the edge $v_1 v_2$ an *auxiliary* edge. After splitting v, every edge of G_P has an edge corresponding to it, and every face of G_P corresponds to a new one in a natural way. Additionally, the dual graph has one new edge, which is dual to $v_1 v_2$.

Let G_S denote the *split planarization* of the outer min-k-planar graph G, that is, the graph G_P with all inner vertices split. See Figure 5 for an example. Observe that there is a one-to-one correspondence between $V(G_P^*)$ and $V(G_S^*)$. Further, every edge of G_P^* has a corresponding edge of G_S^*. The following lemma shows how the properties of a spanning tree T_P of G_P and a spanning tree $\text{dual}(T_P)$ of the dual graph G_P^* are preserved after splitting the vertices of G_P.

▶ **Lemma 11.** *Let G be an expanded outer min-k-planar graph with its split planarization G_S. Then there exists a spanning tree T_S of G_S and a spanning tree $T_S^* = \text{dual}(T_S)$ of G_S^* rooted at f_o^*, such that $\text{depth}(f^*) \leq \lfloor k/2 \rfloor + 1$ for every vertex $f^* \in V(G_S^*)$ and $E(T_S)$ contains all auxiliary edges of G_S.*



(a) The graph G_P with its spanning tree T_P colored black and a spanning tree T_P^* of G_P^* in red.

(b) The graph G_S with its spanning tree T_S colored black and a spanning tree T_S^* of G_S^* in red.

Figure 5 Drawings of an example graphs with their spanning trees and spanning trees of the dual graphs. An example walk, constructed in the proof of Theorem 12, that is connecting w and x, is marked in blue. The vertex f_o^* is missing in both drawings.

Proof. Let G_P be a planarization of G. By Lemmas 9 and 10, there exists a spanning tree T_P^* of G_P^* that is rooted at f_o^* whose vertices have depth at most $\lfloor k/2 \rfloor + 1$. Let $T_P = \text{dual}(T_P^*)$ denote the spanning tree of G_P.

Let G_S denote the graph obtained from G_P by splitting each of its inner vertices. After this transformation, let T_S^* be a tree constructed of edges corresponding to those of T_P^*. Clearly, T_S^* is a spanning tree of the graph G_S^*. The spanning tree $T_S = \text{dual}(T_S^*)$ of G_S contains all auxiliary edges of G_S, because none of the duals of auxiliary edges are in $E(T_S^*)$, as they were not in $E(T_P^*)$. ◀

Now, we are ready to prove the main result of this section.

▶ **Theorem 12.** *Let G be an outer min-k-planar graph. Then* $\text{tw}(G) \leq 3 \cdot \lfloor k/2 \rfloor + 4$.

Proof. By Observation 3 we may assume that G is an expanded outer min-k-planar graph. Let G_S be the split planarization of G. By Lemma 11 there exists a spanning tree T_S of G_S and a spanning tree $T_S^* = \text{dual}(T_S)$ of G_S^* rooted at f_o^* such that $\text{depth}(f^*) \leq \lfloor k/2 \rfloor + 1$ for every vertex $f^* \in V(G_S^*)$ and $E(T_S)$ contains all auxiliary edges of G_S.

We orient the edges of G as follows. Edges incident to the outer face are oriented clockwise, while all other edges are oriented arbitrarily. Observe that every vertex of G has at most two incoming edges.

Now, we construct the tree decomposition $\mathcal{T} = (T_S, B)$ of the graph G. The bags of $\mathcal{T}$ are indexed by the vertices of the spanning tree T_S. We place vertices of G into bags using the following rules.

1. For every outer vertex $v \in V(G_S)$, we place v into the bag B_v.
2. For every oriented edge (x, y) of G, we place x into the bag B_y.
3. For every oriented edge (x, y) of G, and for every inner vertex $v \in V(G_S)$ lying on (x, y), we place x into the bag B_v.
4. For every inner face f and every edge e^* on the path from f^* to f_o^* in T_S^*, where e lies on the edge (x, y) of G, we place x into the bag B_v for each vertex v incident to f.

Note that, in rule 4, the edge e is not in $E(T_S)$, so it is not an auxiliary edge, which implies that it is lying only on a single edge of G.

For every edge (x, y) of G, by rules 1 and 2, the bag B_y contains both x and y. Moreover, every vertex of G is present in some bag. So, to prove that $\mathcal{T}$ is a tree decomposition of G, it suffices to show that for every vertex $x \in V(G)$, the set $\{w : x \in B_w\}$ induces a connected subtree in T_S.

Fix a vertex x and an edge (x, y) of G. Let $e = uv$ be an edge lying on (x, y) such that $e \notin E(T_S)$. Assume that $e^* = f_1^* f_2^*$ and $\text{depth}_{T_S^*}(f_2^*) = \text{depth}_{T_S^*}(f_1^*) + 1$. Let T_e^* be a subtree of T_S^* induced by the set of all descendants of f_2^*, including f_2^*. Define $F_e = \{f : f^* \in V(T_e^*)\}$ and let $\text{boundary}(F_e)$ be the set of vertices incident to some face in F_e. Observe that, by rule 4, x is placed into all bags indexed by $\text{boundary}(F_e)$. The set $\text{boundary}(F_e)$ induces a connected subgraph of T_S, i.e. $T_S[\text{boundary}(F_e)]$ is connected, containing both u and v. Note that the bags of $\mathcal{T}$, into which we placed x by rule 4, are exactly the bags of vertices of T_S that are in $\text{boundary}(F_e)$ for some edge $e \notin E(T_S)$ lying on some edge (x, y) of G.

By rules 1, 2 and 3, x is contained in all bags B_w such that w lies on (x, y) for some edge (x, y) of G. We claim that the vertices indexing these bags, together with the vertices indexing bags we placed x into by rule 4, form a connected subgraph of T_S. To see that, we show that for every vertex w with $x \in B_w$, w is connected to x by a walk in T_S such that bags indexed by the vertices of this walk contain x.

If w lies on an edge (x, y) of G then, in order to construct this walk, we start at vertex w. We iterate over consecutive edges lying on (x, y) between w and x, starting at the edge incident to w. If the current edge e is in $E(T_S)$, then we extend the walk by e. Otherwise, $e \notin E(T_S)$. As $T_S[\text{boundary}(F_e)]$ is connected and every bag of a vertex in $\text{boundary}(F_e)$ contains x, we can extend the walk by some path in $T_S[\text{boundary}(F_e)]$ connecting the endpoints of e. See Figure 5b for an example of such a walk.

If w is in the set $\text{boundary}(F_e)$ for some edge e lying on (x, y), then we begin the walk with a path contained in $\text{boundary}(F_e)$ between w and an endpoint v of e. We extend this walk by a walk between v and x, whose existence we have already proven.

Next, we bound the size of the bags in $\mathcal{T}$. Consider an inner vertex v of T_S. It lies on exactly two edges of G, so by rule 3 we placed two vertices into B_v. Also, v is incident to three non-outer faces of G_S. For every such face f and every edge e^* on the path from f^* to f_o^* in T_S^*, by rule 4 we placed one vertex into B_v. By Lemma 11 we have $\text{depth}(f^*) \leq \lfloor k/2 \rfloor + 1$, so each such path has at most $\lfloor k/2 \rfloor + 1$ edges. Thus, $|B_v| \leq 2 + 3 \cdot (\lfloor k/2 \rfloor + 1)$. Now, consider an outer vertex v of T_S. By rules 1 and 2, the bag B_v contains v and at most two other endpoints of edges incoming to v in G. Also, v is incident to two non-outer faces of G_S. Hence, we derive a bound $|B_v| \leq 3 + 2 \cdot (\lfloor k/2 \rfloor + 1)$. The width of the constructed tree decomposition is at most

$$\max\{2 + 3 \cdot (\lfloor k/2 \rfloor + 1), 3 + 2 \cdot (\lfloor k/2 \rfloor + 1)\} - 1 = 2 + 3 \cdot (\lfloor k/2 \rfloor + 1) - 1 = 3 \cdot \lfloor k/2 \rfloor + 4. \blacktriangleleft$$

5 The separation number of outer min-k-planar graphs

The inequality $\text{sn}(G) \leq \text{tw}(G) + 1$, that bounds the separation number, holds for every graph G. We remark that Theorem 12 directly implies that $\text{sn}(G) \leq 3 \cdot \lfloor k/2 \rfloor + 5$ for every outer min-k-planar graph G. By carefully choosing some bag B_x of a tree decomposition, we can construct a balanced separation (C, D) satisfying $C \cap D = B_x$. To establish a better upper bound, we first prove a general lemma showing how we can obtain a balanced separation (C, D) such that $C \cap D = B_x \cap B_y$ for some two neighboring vertices x, y of a tree decomposition, which needs to satisfy some additional properties.

▶ **Lemma 13.** *Let $\mathcal{T} = (T, B)$ be a tree decomposition of a graph G. Assume that $\Delta(T) \leq 3$ and that every vertex $v \in V(G)$ is in at least two bags of $\mathcal{T}$. Let a be an integer such that $|B_x \cap B_y| \leq a$ for every edge $xy \in E(T)$. Then G has a balanced separation of order at most a.*

Proof. For every edge $xy \in E(T)$, after removing it from T, we obtain two connected components C_x and C_y of T such that $x \in V(C_x)$ and $y \in V(C_y)$. We define $S_{x,y} = \bigcup_{v \in V(C_x)} B_v$ and $S_{y,x} = \bigcup_{v \in V(C_y)} B_v$. It is a well known fact that the pair $(S_{x,y}, S_{y,x})$ is a separation of G of order $|S_{x,y} \cap S_{y,x}| = |B_x \cap B_y| \leq a$.

We claim that there exists an edge $xy \in E(T)$ such that $(S_{x,y}, S_{y,x})$ is a balanced separation of G. Suppose the contrary. Then for every $xy \in E(T)$ it holds that $|S_{x,y} \setminus S_{y,x}| > \frac{2}{3}n$ or $|S_{y,x} \setminus S_{x,y}| > \frac{2}{3}n$, where $n = |V(G)|$. Now, we orient every edge of T. If the first inequality holds then we orient xy as (y, x), in the other case as (x, y). Also, note that $|S_{x,y} \setminus S_{y,x}| > \frac{2}{3}n$ implies $|S_{y,x}| < \frac{1}{3}n$.

The tree T with oriented edges is an acyclic graph, so there exists a sink, that is, a vertex in T such that all edges incident to x are oriented towards x. Let $\{y_1, \ldots, y_d\}$, where $d \leq 3$, be the set of neighbors of x in T. We have $|S_{y_i,x}| < \frac{1}{3}n$. Moreover,

$$\bigcup_{1 \leq i \leq d} S_{y_i,x} = \bigcup_{v \in V(T) \setminus \{x\}} B_v = V(G),$$

since every vertex of G is in at least two bags of $\mathcal{T}$. We obtain the following inequalities

$$|V(G)| = \left| \bigcup_{1 \leq i \leq d} S_{y_i,x} \right| \leq \sum_{1 \leq i \leq d} |S_{y_i,x}| < d \cdot \frac{1}{3}n \leq n,$$

which is a contradiction. ◀

Now, we are ready to establish an upper bound on the separation number of outer min-k-planar graphs.

▶ **Theorem 14.** *Let G be an outer min-k-planar graph. Then* $\mathrm{sn}(G) \leq 2 \cdot \lfloor k/2 \rfloor + 4$.

Proof. The class of outer min-k-planar graphs is closed under taking subgraphs. Therefore, it suffices to find a balanced separation of order at most $2 \cdot \lfloor k/2 \rfloor + 4$ for every maximal outer min-k-planar graph G. Let H be an expanded outer min-k-planar graph obtained from G by Observation 3. By Theorem 12, there exists a tree decomposition $\mathcal{T} = (T_S, B)$ of H, where T_S is a spanning tree of the split planarization of H. From the proof of Theorem 12, it follows that $\Delta(T_S) \leq 3$ and that every vertex $v \in V(H)$ is in at least two bags of $\mathcal{T}$ (since there is an oriented edge (v, w) in H, so $v \in B_v$ and $v \in B_w$).

We construct a tree decomposition $\mathcal{T}' = (T_S, B')$ of G with $B'_x = \{\mathrm{org}(v) : v \in B_x\}$, where $\mathrm{org}(v)$ denotes the original vertex that v replaced in the transformation described in Observation 3. Every vertex $v \in V(G)$ is in at least two bags of $\mathcal{T}'$, since every image of v is in at least two bags of $\mathcal{T}$. Every edge $vw \in E(G)$ is realized in some bag of $\mathcal{T}'$, because in H there is an edge corresponding to vw between an image of v and an image of w. To prove that, for every vertex v of G, the bags of $\mathcal{T}'$ containing v are spanning a connected subtree of T_S, we denote the images of v by $v_1, \ldots, v_s$, ordered along the outer face. Since H is maximal, for every $i \in \{1, \ldots, s-1\}$, there is an edge $v_i v_{i+1}$ in $E(H)$. Thus, the two subtrees of T_S induced by the bags of $\mathcal{T}$ containing v_i and those containing v_{i+1} share a common vertex. Bags containing v in $\mathcal{T}'$ are spanning a connected subtree of T_S, because this subtree is a union of subtrees spanned by the images of v. Hence, $\mathcal{T}'$ is a valid tree decomposition of G.

We say that a vertex v was placed into a bag B_x of $\mathcal{T}$ due to rule 4 of constructing the tree decomposition being applicable to the vertex v and a face f if:

- the face f is incident to x;
- there exists an edge e^* on the path between f_o^* and f^* in T_S^* such that e lies on an edge (v, w) of H, for some $w \in V(H)$.

Now, we want to show that, for every edge $xy \in E(T_S)$, we have $\left|B'_x \cap B'_y\right| \leq 2 \cdot \lfloor k/2 \rfloor + 4$. Let f_1 and f_2 be the faces of H_S incident to xy.

▷ Claim. If $v \in B'_x \cap B'_y$ then there exists an image v_t of v such that either

- xy lies on an edge (v_t, w) of H, for some $w \in V(H)$; or
- v_t was placed into both B_x and B_y due to rule 4 of constructing $\mathcal{T}$ being applicable to v_t and face f_1 or face f_2.

Proof. If x has degree 3 in H_S, then let $f_x \notin \{f_1, f_2\}$ be the face of H_S incident to vertex x. Similarly, if y has degree 3 in H_S, then let $f_y \notin \{f_1, f_2\}$ be the face of H_S incident to y. Assume that $v \in B'_x \cap B'_y$, but no image of v was placed into B_x and B_y due to the reasons stated in the claim. So there exist v_{i_x} and v_{i_y} that are, not necessarily distinct, images of v such that $v_{i_x} \in B_x$ and $v_{i_y} \in B_y$. For $t \in \{x, y\}$, vertex v_{i_t} was placed into B_t because either

1. t lies on an edge (v_{i_t}, w_t) of H such that xy does not lie on (v_{i_t}, w_t); or
2. when t has degree 3 in H_S, rule 4 of constructing $\mathcal{T}$ is applicable to the vertex v_{i_t} and face f_t, i.e. there exists an edge (v_{i_t}, w_t) of H and an edge e_t of H_S lying on (v_{i_t}, w_t) such that e_t^* is on the path between f_t^* and f_o^* in T_S^*.

Now, we draw a curve $\mathcal{C}$ on the drawing of H_S. The curve $\mathcal{C}$ consists of the drawing of the edge xy and the drawing of an arc of the outer face between v_{i_x} and v_{i_y} that contains only images of v (the images of v are spanning a single arc of the outer face). Next, we add to $\mathcal{C}$ curves connecting v_{i_x} with x and v_{i_y} with y. For $t \in \{x, y\}$, the way we draw these curves depends on the reason v_{i_t} was placed into B_t, considered in the same order as above.

1. If t lies on an edge (v_{i_t}, w_t) of H, we draw along (v_{i_t}, w_t), starting at vertex v_{i_t} and ending at vertex t.
2. If rule 4 of constructing $\mathcal{T}$ is applicable to the vertex v_{i_t} and face f_t, let p_t be a path between f_t^* and f_o^* in T_S^*. We draw along (v_{i_t}, w_t), starting at vertex v_{i_t} and ending at the crossing point with the drawing of p_t. We continue along p_t till vertex f_t^*. Finally, we connect vertices f_t^* and t with a segment.

Note that if $x = v_{i_x}$ and $y = v_{i_y}$, then $\mathcal{C}$ is degenerated to the arc between v_{i_x} and v_{i_y}, implying that xy connects two consecutive images of v – contradiction. Otherwise, we claim that one of the closed regions induced by $\mathcal{C}$ contains f_1 or f_2. Indeed, $\mathcal{C}$ follows edges of H_S and edges of T_S^*, but cannot contain f_1^* nor f_2^*, because then rule 4 of constructing $\mathcal{T}$ would be applicable to vertex v_{i_x} or v_{i_y} and face f_1 or f_2. The segments between f_t^* and t does not intersect f_1 nor f_2. We may assume that f_1 is contained inside a closed region induced by $\mathcal{C}$. Consider a path p_1 between f_1^* and f_o^* in T_S^*. Since f_1^* is inside $\mathcal{C}$ and f_o^* is outside $\mathcal{C}$, drawing of p_1 has to intersect $\mathcal{C}$. We consider where the first intersection point is located.

- Path p_1 cannot intersect e nor the segments between f_t^* and t.
- If p_1 intersects an edge (v_{i_t}, w_t) then rule 4 of constructing $\mathcal{T}$ is applicable to v_{i_t} and f_1.
- If p_1 intersects p_t, then p_1 follows along p_t up to the intersection point with (v_{i_t}, w_t), so the previous case applies.
- If p_1 intersects the arc of the outer face between v_{i_x} and v_{i_y} then it has to intersect an edge (v_r, v_{r+1}), where v_r and v_{r+1} are consecutive images of v on the outer face. Hence, rule 4 of constructing $\mathcal{T}$ is applicable to v_r and f_1.

In all cases, we obtain a contradiction. ◁

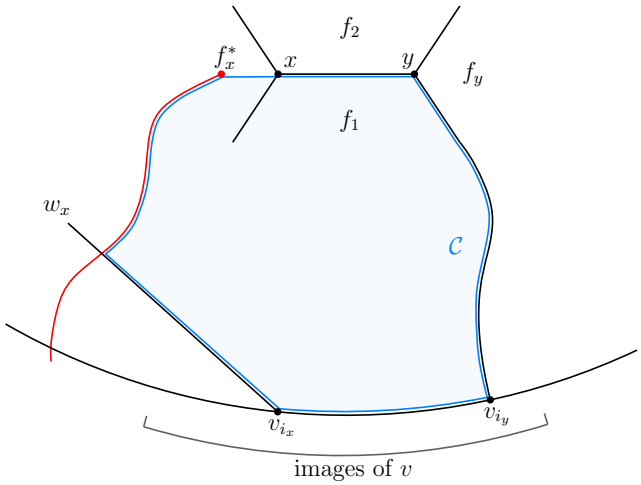


Figure 6 Drawing of an example curve $\mathcal{C}$.

We proved that if $v \in B'_x \cap B'_y$ then there is an image v_t of v such that either

- xy lies on an edge (v_t, w) of H, for some $w \in V(H)$; or
- v_t was placed into both B_x and B_y due to rule 4 of constructing the tree decomposition $\mathcal{T}$ being applicable to vertex v_t and face f_1 or f_2.

Note that xy lies on at most two edges of H (two if xy is an auxiliary edge, one otherwise). Moreover, the two paths from f_o^* to f_1^* and from f_o^* to f_2^* in T_S^* each have at most $\lfloor k/2 \rfloor + 1$ edges. Therefore, $\left|B'_x \cap B'_y\right| \leq 2 + 2 \cdot (\lfloor k/2 \rfloor + 1) = 2 \cdot \lfloor k/2 \rfloor + 4$. By applying Lemma 13 to $\mathcal{T}'$, we obtain that G has a balanced separation of order at most $2 \cdot \lfloor k/2 \rfloor + 4$. ◀

To give a lower bound we define a graph called *stacked prism*. A stacked prism $Y_{m,n}$ is an $m \times n$ grid with additional edges connecting the vertices of the first and the last row that are in the same column. The $Y_{m,n}$ has an outer $(2n-2)$-planar drawing, thus also an outer min-$(2n-2)$-planar drawing. In the cyclic order of the drawing, we place rows consecutively, one after another. The edges from rows cross no other edges and the edges from columns cross exactly $2n-2$ other edges. The authors of [7] showed that for every number n and for every sufficiently large even number m, $\mathrm{sn}(Y_{m,n}) = 2n$. This leads to the following theorem.

▶ **Theorem 15.** *For every even number k, there exists an outer min-k-planar graph G such that* $\mathrm{sn}(G) = k + 2$.

We remark that the multiplicative constant of 1 in the upper bound given in Theorem 14 is tight, as it matches that of the lower bound in Theorem 15.

References

1 Michael J. Bannister and David Eppstein. Crossing minimization for 1-page and 2-page drawings of graphs with bounded treewidth. In Christian A. Duncan and Antonios Symvonis, editors, *22nd International Symposium on Graph Drawing (GD 2014)*, volume 8871 of *Lecture Notes in Computer Science*, pages 210–221. Springer, 2014. doi:10.1007/978-3-662-45803-7_18.

2 Michael J. Bannister and David Eppstein. Crossing minimization for 1-page and 2-page drawings of graphs with bounded treewidth. *Journal of Graph Algorithms and Applications*, 22(4):577–606, 2018. doi:10.7155/jgaa.00479.

3 Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1-2):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.

4 Steven Chaplick, Myroslav Kryven, Giuseppe Liotta, Andre Löffler, and Alexander Wolff. Beyond outerplanarity. In Fabrizio Frati and Kwan-Liu Ma, editors, *25th International Symposium on Graph Drawing and Network Visualization (GD 2017)*, volume 10692 of *Lecture Notes in Computer Science*, pages 546–559. Springer, 2017. doi:10.1007/978-3-319-73915-1_42.

5 Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

6 Zdeněk Dvořák and Sergey Norin. Treewidth of graphs with balanced separations. *Journal of Combinatorial Theory, Series B*, 137:137–144, 2019. doi:10.1016/j.jctb.2018.12.007.

7 Oksana Firman, Grzegorz Gutowski, Myroslav Kryven, Yuto Okada, and Alexander Wolff. Bounding the treewidth of outer k-planar graphs via triangulations. In *GD 2024: 32nd International Symposium on Graph Drawing and Network Visualization*, volume 320 of *LIPIcs*, pages 14:1–14:17, 2024. doi:10.4230/LIPIcs.GD.2024.14.

8 Hussein Houdrouge, Babak Miraftab, and Pat Morin. Separation number and treewidth, revisited, 2025. doi:10.48550/arXiv.2503.17112.

9 Frank Kammer and Torsten Tholey. A lower bound for the treewidth of k-outerplanar graphs. Technical report, Fakultät für Angewandte Informatik, Universität Augsburg, 2009. URL: https://opus.bibliothek.uni-augsburg.de/opus4/frontdoor/index/index/docId/1304.

10 Yasuaki Kobayashi, Yuto Okada, and Alexander Wolff. Recognizing 2-Layer and Outer k-Planar Graphs. In *41st International Symposium on Computational Geometry (SoCG 2025)*, volume 332 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 65:1–65:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.SoCG.2025.65.

11 Neil Robertson and Paul D. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986. doi:10.1016/0196-6774(86)90023-4.

12 Paul D. Seymour and Robin Thomas. Graph searching and a min-max theorem for tree-width. *Journal of Combinatorial Theory, Series B*, 58(1):22–33, 1993. doi:10.1006/jctb.1993.1027.

13 David R. Wood and Jan Arne Telle. Planar decompositions and the crossing number of graphs with an excluded minor. *New York Journal of Mathematics*, 13:117–146, 2007. URL: https://nyjm.albany.edu/j/2007/13-8.html.

Stabbing Faces by a Convex Curve

David Eppstein
Computer Science Department, University of California, Irvine, CA, USA

Abstract

We prove that, for every plane graph G and every smooth convex curve C not on a single line, there exists a straight-line drawing of G for which every face is crossed by C.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases planar graphs, convex curves, stabbing, transversal

Digital Object Identifier 10.4230/LIPIcs.GD.2025.29

Related Version *Full Version*: https://arxiv.org/abs/2508.17549

Funding *David Eppstein*: Research supported in part by NSF grant CCF-2212129.

1 Introduction

Every smooth convex curve C, not on a single line, contains arbitrarily many points in general position (its tangent points at distinct slopes within a suitable interval). It follows that every outerplanar graph has a straight-line drawing with all vertices on C [10]. A graph that can be drawn in this way must be outerplanar, as its vertices all belong to a common face outside C. But instead of asking for C to contain all vertices, here we ask: which planar graphs can be drawn so that C intersects all faces? The answer turns out to be all of them.

Our motivation for this question comes from the problem of characterizing universal point sets, sets of points U_n for each n having the property that every n-vertex planar graph has a straight-line drawing with its vertices in U_n. These sets have size $O(n^2)$ [1, 3, 6, 16], and $\geq cn$ for a constant $c > 1$ [5, 13, 15]. A recent line of research has investigated the combinatorial structure of universal point sets, in terms of the number of lines necessary to cover all points of such a set. Although point sets on $O(1)$ parallel lines can support only the planar graphs of bounded pathwidth [7], two crossed lines can support a much wider class of graphs, including all leveled planar graphs [2]. However, supporting all n-vertex planar graphs requires an unbounded number of lines, $\Omega(n^{1/3})$ [4, 9, 14]. Eppstein [8] asked: what if we replace lines in these results by convex curves? Is it possible that universal point sets can be supported by only $O(1)$ convex curves?

This question could be answered by finding a single planar graph $G^?$ that could not be drawn with all of its faces crossed by a single convex curve. If such a graph $G^?$ existed, we could assume without loss of generality that $G^?$ is a triangulation, and then recursively replace triangular faces of $G^?$ by more copies of $G^?$. This recursive replacement process would generate a family of graphs in which, for every drawing, every convex curve touches a number of faces bounded by a sublinear polynomial of the number of vertices. This would imply that universal point sets for graphs in this family could be supported only by a polynomially growing number of convex curves. However, the hope for a proof along these lines is dashed by our result: $G^?$ does not exist.

We remark that the corresponding question for edges rather than faces has an intermediate answer: the class of graphs that can be drawn with all edges touching a smooth convex curve C is broader than the outerplanar graphs but less broad than the planar graphs. In the full version of this paper, we provide an example of a planar graph that cannot be drawn in this way, regardless of the choice of C. It takes the form of the graph of nine regular octahedra,


© David Eppstein;
licensed under Creative Commons License CC-BY 4.0

with the outer eight octahedra each glued to a separate face of a central octahedron. But unlike the case for faces crossed by C, we do not know how to leverage this example to prove anything about the number of convex curves needed to support a universal point set.

2 Preliminaries

Given a plane graph G,[1] to be drawn with all faces crossed by a convex curve, we may assume without loss of generality that G is maximal planar, because if we complete any other graph to a maximal planar graph, and draw the result with C crossing all faces of the maximal planar completion, then C will also cross all faces of the original graph.

For a maximal plane graph G, with a fixed choice of outer face, we will draw G using a *canonical ordering* [12], as used by de Fraysseix, Pach, and Pollack to find grid drawings of planar graphs [6]. This is an ordering of the vertices of G such that the first three vertices in the ordering induce a triangle with one edge (the *base edge*) on the outer face, and such that each prefix of the ordering induces a triangulated disk[2] in the drawing of G, with the bounded faces of this triangulated disk also being faces of the full drawing of G. The next vertex in the ordering, after any prefix, must have as its earlier neighborhood a contiguous path along the boundary of the disk, not including the base edge, and the region between this path and the next vertex is triangulated in a fan of triangles meeting at the added vertex. This structure lends itself well to greedy incremental algorithms for planar graph drawing and to induction proofs of properties of these drawings.

For every positive integer k, there exist maximal planar graphs in which, for every drawing, the number of faces of the graph that can be crossed by a line is less than a $1/k$ fraction of all faces [11]. These graphs therefore cannot have all faces crossed by a k-gon. To avoid these examples, our results involve smooth convex curves. There are many ways of defining curves and their smoothness; the properties we require are that the curve must form a connected subset of the boundary of a convex set in the plane (for instance, its convex hull), it must have a unique tangent line at each point, and the slope of the tangent line must be continuous along the curve (C^1 smoothness, meaning that the first derivative is continuous). We do not require strict convexity: our curves may contain line segments (along which the tangent slope is constant). For instance, a *stadium*, a convex shape obtained by attaching semicircular end caps to two opposite sides of a rectangle, has a smooth convex boundary in this sense. However, a polygon is not smooth at its vertices. An *arc* of a convex curve is any connected proper subset. The *total curvature* of an arc can be measured by the change in slopes of the tangent lines along the arc, as an angle.

With a stronger smoothness assumption of being twice continuously differentiable (C^2 smoothness), the ability to cross all faces of a planar drawing would extend to non-convex smooth curves that do not lie on a line, because each such a curve has a smooth convex arc (within a neighborhood of any point of nonzero curvature) to which we can apply our proof. We omit the details.

---

[1] By a *plane graph* we mean a planar graph together with a combinatorial embedding, which should be respected in our drawing of the graph.

[2] By a *triangulated disk*, we mean a plane graph in which the outer face is a simple polygon and all interior faces are triangles.

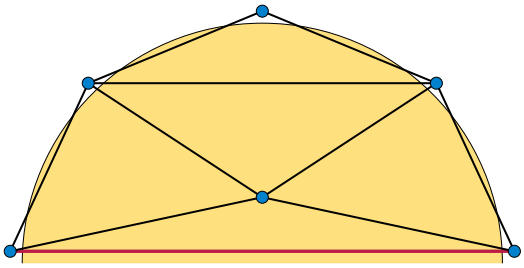


Figure 1 Adequate drawing of a triangulated disk (base edge shown in red) with respect to a semicircle: all boundary vertices are exterior to the semicircle and all non-base boundary edges are crossed, twice each, by disjoint arcs of the semicircle.

3 Adequate drawings

We will separate out into lemmas the steps of an induction proof that a given plane graph G has a drawing with all faces crossed by a given smooth convex curve C. If we reinterpret the proof as a greedy construction algorithm, each step of the construction has the following property.

▶ **Definition 1.** *Given an abstract triangulated disk D with a specified base edge B, and a smooth convex curve C that is not on a single line, we define an* adequate drawing *of D to be a straight-line drawing with the following properties:*

- *Every boundary vertex of D is exterior to the convex hull of C.*
- *Every boundary edge e of D other than B is crossed twice by C. The two crossing points bound an arc of C that is disjoint from D and from the other arcs defined in the same way from other boundary edges of D.*
- *The crossings of C with D span an arc of C that bends through a total curvature of less than π.*

Figure 1 depicts an example. Not all faces in this example are crossed by the convex curve C (a semicircle), but the faces incident to the non-base boundary edges are crossed. As the name suggests, it will be adequate to prove that every triangulated disk formed by the canonical ordering has an adequate drawing, as every face will be adjacent to the boundary for the induced disk at some point in the ordering. This ensures the property we are really trying to prove, that the whole drawing of G has every face crossed by C:

▶ **Lemma 2.** *Let G be a maximal plane graph G with n vertices, with a straight-line drawing in the plane, and let C be a smooth curve that is not on a single line. Suppose that G has a canonical ordering with the property that, for each triangulated disk D_i (with $3 \le i \le n$) induced by the first i vertices in the canonical ordering, the drawing of G restricted to D_i is adequate. Then the drawing of G has all faces crossed by C.*

Proof. Each face Δ of G has at least one edge e_Δ that is a boundary edge of some D_i, but not the base edge of the canonical ordering. For the outer face, e_Δ can be any non-base edge of D_n; for any other face, e_Δ is the edge of Δ induced by the first two vertices of Δ to appear in the canonical ordering. Because it is a non-base boundary edge of D_i, and because of the assumption that the drawing of D_i is adequate, e_Δ is crossed (twice) by C. Near these edge crossing points, Δ itself is crossed by C. ◀

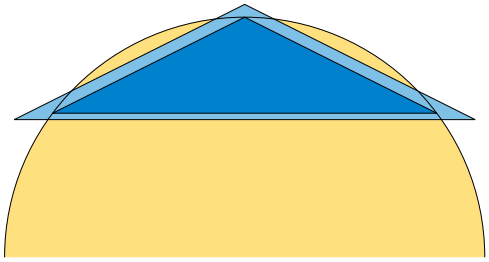


Figure 2 Base case (Lemma 3): constructing an adequate drawing of a triangle by scaling an inscribed triangle with three distinct tangent slopes.

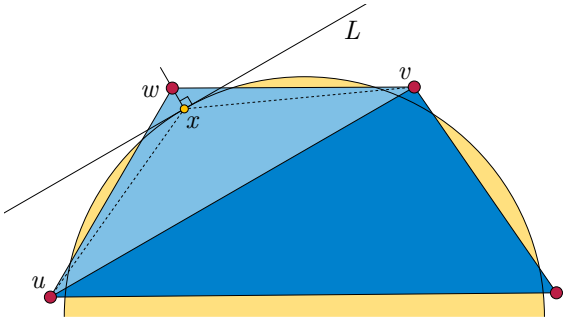


Figure 3 Ear case (Lemma 4): attaching triangle uvw to edge uv of an adequate drawing.

4 Induction proof

We will prove our main result by induction, using canonical orderings. The next lemma is the base case for this induction proof.

▶ **Lemma 3.** *For every smooth convex curve C that does not lie on a line, there exists an adequate drawing of a triangle with a designated base edge.*

Proof. Let A be any arc of C of nonzero curvature less than π, and inscribe a triangle in A so that the tangent lines to A at its three vertices have distinct slopes; this is possible because, by smoothness and nonzero curvature, A has tangents with a continuous range of slopes. Choose the base edge of this triangle to be the edge connecting the two extreme vertices in their ordering along A. Scale the triangle around its centroid by a factor $1 + \varepsilon$ for a parameter $\varepsilon > 0$ chosen sufficiently small,

By convexity, the inscribed triangle lies in the closed convex hull of C, and no point in the interior of one of its edges can lie on curve C itself: if such a point existed then the convexity of C would force C to contain that edge, violating the assumption of distinct tangent slopes at the three vertices. Scaling brings the triangle vertices exterior to C, but for sufficiently small choices of ε, some points interior to each non-base edge remain interior to the convex hull of C (Figure 2). Then each non-base edge of the scaled triangle will have two crossings with C separating its endpoints (exterior to the hull) with the points along the edge interior to the hull. These crossings will separate disjoint arcs of C, and together span an arc of C that is a sub-arc of A, so all the requirements of an adequate drawing are met. ◀

The simpler of the two inductive cases involves adding a single triangle as an ear of the triangulated disk, attached to it by a single boundary edge of the previous disk.

▶ **Lemma 4.** *Let D be a triangulated disk with an adequate drawing with respect to a smooth convex curve C, and let uv be any boundary edge of the drawing of D that is not the base edge of the drawing. Then it is possible to add a vertex w adjacent only to u and v, forming a triangle attached to D across edge uv, and to place w in the drawing so that it becomes an an adequate drawing of the resulting augmented triangulated disk.*

Proof. For the notation used here, see Figure 3. By the assumption that the drawing of D is adequate, an arc of curve C crosses twice through edge uv. By smoothness, this arc has a tangent line L parallel to uv, touching the arc at a point x between the two crossings of C with uv. Polyline uxv starts outside C at u, crosses inside C to touch C at x from the interior of its convex hull, and then crosses C again to reach the point v exterior to C. Place w on a perpendicular line to L through x, outward from C at a sufficiently small distance ε from x.

If ε is sufficiently small, the polyline uwv crosses C from exterior to interior at a point near the crossing of C with line segment ux, and crosses again from interior to exterior at a point near the crossing of C with line segment xv. In order for polyline uwv to be exterior to C at w, between these two crossings, each of the two new edges uw and wv must have a second crossing near w. Thus, the requirements of an adequate drawing at the new boundary edges uw and wv are met, and the crossings of C with the other boundary edges of D remain unaffected, so the result is an adequate drawing. ◀

In the remaining inductive case, we add a fan of triangles, connecting an added vertex to a path of two or more edges of the triangulated disk.

▶ **Lemma 5.** *Let D be a triangulated disk with an adequate drawing with respect to a smooth convex curve C, and let $P = v_0v_1 \dots v_k$ be a path of k boundary edges of the drawing of D (for $k \geq 2$) that does not pass through the base edge of the drawing. Then it is possible to add a vertex w adjacent to each vertex v_i of the path, forming a fan of triangles attached to P across edge uv, and to place w in the drawing so that it becomes an adequate drawing of the resulting augmented triangulated disk.*

Proof. For the notation used here, see Figure 4. Draw a ray from vertex v_0 through vertex v_1, and a second ray from vertex v_k through vertex v_{k-1}. Each ray crosses C at the two crossing points of an edge of P. By the assumption that the drawing is adequate and therefore its crossings with C span an arc of C that bends through an angle less than π, the two rays cross rather than diverging. Let x be the crossing point of the two rays. At x, the two crossing rays subdivide the plane into four wedges. Place w interior to the wedge opposite from P, at a sufficiently small distance from w.

Vertices $v_1, \dots v_{k-1}$ all lie in the wedge containing P, and are visible to x with respect to the given drawing of D; however, the two endpoints v_0 and v_k of the path are blocked from visibility by v_1 and v_{k-1} respectively. Placing w anywhere in the wedge opposite from P preserves the visibilities with $v_1, \dots v_{k-1}$ and allows w to also see the two path endpoints, producing a valid drawing of D. The segments v_0x and xv_k cross C twice, along edges v_0v_1 and $v_{k-1}v_k$ (which lie on these segments), and if w is sufficiently close to x then the segments v_0w and wv_k will also cross C twice, meeting the requirements of an adequate drawing at the two new boundary edges of the augmented drawing. The crossings of C with the other boundary edges of D remain unaffected, so the result is an adequate drawing. ◀

With the lemmas above, we are ready to prove our main result:

▶ **Theorem 6.** *Let G be a plane graph, and let C be a smooth convex curve that does not lie on a line. Then G has a straight-line drawing in which all faces are crossed by C.*

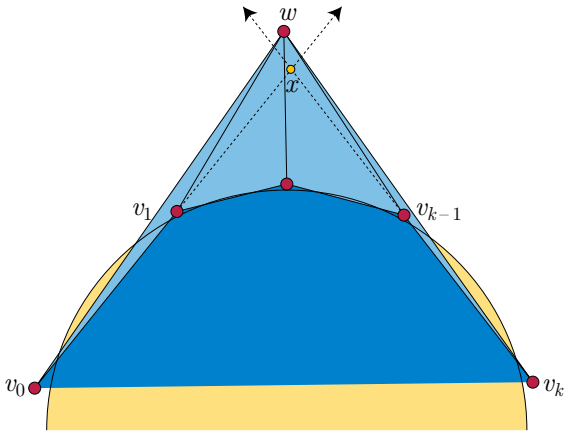


Figure 4 Fan case (Lemma 5): attaching a fan of triangles to path $P = v_0 v_1 \dots v_k$ of an adequate drawing.

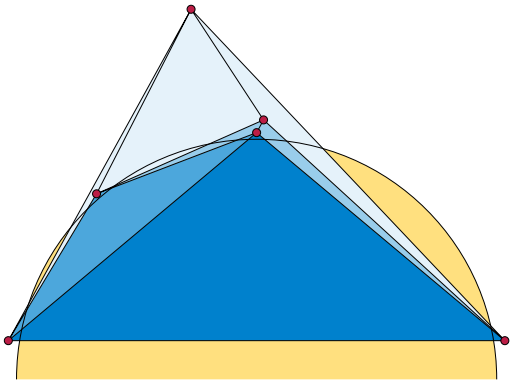


Figure 5 Drawing of the graph of an octahedron with all faces crossed by a semicircle. The shading indicates the order in which triangles were added to the drawing in its canonical ordering, with darker triangles added earlier than lighter ones.

Proof. Augment G to a maximal plane graph, and construct a canonical ordering. We prove by induction that there exists a drawing D of the augmented graph such that, for each triangulated disk D_i (with $3 \leq i \leq n$) induced by the first i vertices in the canonical ordering, the drawing restricted to D_i is adequate. The base case $i = 3$ is Lemma 3, and the induction step for each $i > 4$ is either Lemma 4 (if vertex i has two neighbors in D_i) or Lemma 5 (if vertex i has three or more neighbors in D_i. This proves that D meets the hypotheses of Lemma 2, so every face of the augmented graph is crossed by C. Each face of G itself is a union of faces of the augmented graph, so it is also crossed by C. ◀

Our induction proof can be turned into an algorithm in which we add vertices one at a time to a drawing, using the construction of Lemma 3 to add the first three vertices, and either Lemma 4 or Lemma 5 to add each subsequent ear. An example of the resulting drawing, for the graph of an octahedron with its faces crossed by a semicircle, is shown in Figure 5. However, to analyze the numerical precision of vertex coordinates needed for this construction, and the angular resolution of the result, we would need additional assumptions about C. As the figure shows, even for a well-behaved curve such as a semicircle, the angular resolution can be quite low: this construction is more useful in producing counterexamples than as a method for producing readable graph drawings.

5 Conclusions and open problems

We have shown that for every planar graph and every smooth convex curve, there exists a drawing of the graph in which all faces are crossed by the curve. The most salient remaining open problem is the one that motivated this research: is there a constant k such that all planar graphs can be drawn with their vertices on $\leq k$ convex curves? It would also be of interest to more precisely characterize the graphs that can be drawn with all edges crossed by a convex curve (for which see the full version of this paper) and to extend these results to beyond-planar graphs.

References

1 Michael J. Bannister, Zhanpeng Cheng, William E. Devanny, and David Eppstein. Superpatterns and universal point sets. *J. Graph Algorithms Appl.*, 18(2):177–209, 2014. Preliminary version in *21st Int. Symp. Graph Drawing*, 2013, LNCS 8242, pp. 340–351. doi:10.7155/jgaa.00318.

2 Michael J. Bannister, William E. Devanny, Vida Dujmović, David Eppstein, and David R. Wood. Track layouts, layered path decompositions, and leveled planarity. *Algorithmica*, 81(4):1561–1583, 2019. Preliminary version in *24th Int. Symp. Graph Drawing*, 2016, LNCS 9801, pp. 499–510. doi:10.1007/s00453-018-0487-5.

3 Franz J. Brandenburg. Drawing planar graphs on $\frac{8}{9}n^2$ area. In *Proc. Int. Conf. Topological and Geometric Graph Theory*, volume 31 of *Electron. Notes Discrete Math.*, pages 37–40. Elsevier, 2008. doi:10.1016/j.endm.2008.06.005.

4 Steven Chaplick, Krzysztof Fleszar, Fabian Lipp, Alexander Ravsky, Oleg Verbitsky, and Alexander Wolff. Drawing graphs on few lines and few planes. *J. Comput. Geom.*, 11(1):433–475, 2020. Preliminary version in *15th Int. Symp. Algorithms and Data Structures*, 2017, LNCS 10389, pp. 265–276. doi:10.20382/jocg.v11i1a17.

5 M. Chrobak and T. H. Payne. A linear-time algorithm for drawing a planar graph on a grid. *Inform. Process. Lett.*, 54(4):241–246, 1995. doi:10.1016/0020-0190(95)00020-D.

6 H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990. doi:10.1007/BF02122694.

7 Vida Dujmović, Michael R. Fellows, Matthew Kitching, Giuseppe Liotta, Catherine McCartin, Naomi Nishimura, Prabhakar Ragde, Frances Rosamond, Sue Whitesides, and David R. Wood. On the parameterized complexity of layered graph drawing. *Algorithmica*, 52(2):267–292, 2008. Preliminary version in *9th European Symp. Algorithms*, 2001, LNCS 2161, pp. 488–499. doi:10.1007/s00453-007-9151-1.

8 David Eppstein. *Forbidden Configurations in Discrete Geometry*. Cambridge University Press, 2018. Open Problem 16.14, p. 187. doi:10.1017/9781108539180.

9 David Eppstein. Cubic planar graphs that cannot be drawn on few lines. *J. Comput. Geom.*, 12(1):178–197, 2021. Preliminary version in *35th Int. Symp. Computational Geometry*, 2019, LIPIcs 129, 32:1–32:15. doi:10.20382/v12i1a8.

10 P. Gritzmann, B. Mohar, J. Pach, and R. Pollack. Embedding a planar triangulation with vertices at specified points (solution to problem E3341). *American Mathematical Monthly*, 98(2):165–166, 1991. doi:10.2307/2323956.

11 Branko Grünbaum and Hansjoachim Walther. Shortness exponents of families of graphs. *J. Combinatorial Theory Ser. A*, 14:364–385, 1973. doi:10.1016/0097-3165(73)90012-5.

12 G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16(1):4–32, 1996. doi:10.1007/s004539900035.

13 Maciej Kurowski. A 1.235 lower bound on the number of points needed to draw all n-vertex planar graphs. *Inform. Process. Lett.*, 92(2):95–98, 2004. doi:10.1016/j.ipl.2004.06.009.

14 Alexander Ravsky and Oleg Verbitsky. On collinear sets in straight-line drawings. In Petr Kolman and Jan Kratochvíl, editors, *Proc. 37th Int. Worksh. Graph-Theoretic Concepts in*

Computer Science (WG 2011), volume 6986 of *Lecture Notes in Computer Science*, pages 295–306. Springer, 2011. doi:10.1007/978-3-642-25870-1_27.

15 Manfred Scheucher, Hendrik Schrezenmaier, and Raphael Steiner. A note on universal point sets for planar graphs. *J. Graph Algorithms Appl.*, 24(3):247–267, 2020. Preliminary version in *27th Int. Symp. Graph Drawing*, 2019, LNCS 11904, pp. 350–362. doi:10.7155/jgaa.00529.

16 Walter Schnyder. Embedding planar graphs on the grid. In David S. Johnson, editor, *Proc. 1st ACM–SIAM Symp. on Discrete Algorithms (SODA 1990)*, pages 138–148. SIAM, 1990. URL: https://dl.acm.org/citation.cfm?id=320176.320191.

Universal Quality Metrics for Graph Drawings: Which Graphs Excite Us Most?

Gavin J. Mooney
Monash University, Melbourne, Australia

Tim Hegemann
Universität Würzburg, Germany

Alexander Wolff
Universität Würzburg, Germany

Michael Wybrow
Monash University, Melbourne, Australia

Helen C. Purchase
Monash University, Melbourne, Australia

Abstract

Graphs are drawn for various purposes, and drawings are meant to display various features of a graph (such as planarity, Hamiltonicity). Still, there is a long history in measuring the quality of a graph drawing. Most of the metrics that have been implemented and used in large studies assume that graphs are drawn straight-line. Most of the studies use randomly generated graphs or one of very few existing benchmark sets that consist of graphs with a specific technical background (e.g., telecommunication networks).

In this paper, we extend ten commonly used metrics to node-link diagrams where edges can be curves or polygonal chains. We implement these measures and use them to evaluate a new collection of graph drawings that we have extracted from 27 proceedings of the Graph Drawing conference using an automated pipeline. We compare the "metrics landscape" of our new benchmark set, the *GD-collection-v1*, which seems to mostly contain manually drawn graphs, to the metric landscape of a benchmark set with randomly generated graphs and computer-generated straight-line drawings that has been used in a recent study [Mooney et al.; PacificVis 2024].

Comparing the *GD-collection-v1* with the Mooney at al. dataset reveals a distinct metrics landscape: GD drawings come from much smaller graphs (median vertex number 11 vs. 48) and therefore attain higher medians on most readability metrics. For example, Neighbourhood Preservation (0.5 vs. 0.239) is markedly higher in the *GD-collection-v1*. We also find that a large proportion of extracted drawings contain curved and/or polygonal edges (57%), motivating the extended metric definitions.

2012 ACM Subject Classification Human-centered computing → Graph drawings

Keywords and phrases Graph drawing metrics, metric landscape, straight-line drawings, polyline drawings, curved drawings, automated extraction of graph drawings

Digital Object Identifier 10.4230/LIPIcs.GD.2025.30

Supplementary Material *Dataset*: `https://github.com/hegetim/gd-collection` [16]
archived at `swh:1:dir:478a27dd277dc5818bdf699d2a5bc222a010533b`
Software (Source Code): `https://github.com/gavjmooney/geg` [17]
archived at `swh:1:dir:91f45ae7976a74b00a0bf86145b52dd78838fb29`

Funding *Tim Hegemann*: Supported by BMFTR grant 01IS22012C.

Acknowledgements We thank Sebastian Kempf for his contributions to Graph Harvester.

© Gavin J. Mooney, Tim Hegemann, Alexander Wolff, Michael Wybrow, and Helen C. Purchase;
licensed under Creative Commons License CC-BY 4.0

1 Introduction

Graph drawings are typically valued for the extent to which they conform to well-established layout principles (sometimes called "aesthetics" in the literature), for example, minimising the number of edge crossings, minimising the "stress" or maximising the angle between edges incident to the same node. While many of these principles have been used over several years of graph drawing research, they have not all (or always) been defined consistently; for example stress [25]. Recently, Mooney et al. [18] defined ten common principles as "unambiguous metrics" – some defined mathematically, some algorithmically, and all normalised to lie between 0 and 1, where 1 describes the principle in the form that it is assumed best supports human understanding. These metrics apply to connected, straight-line drawings of graphs. The authors also defined a "metrics landscape" that shows the distribution of each of the metric values over almost half a million graph drawings. These graph drawings were created by applying six well-known layout methods (and a random layout) to almost 17,000 randomly generated graphs, five times.

In this work, we generalise the above metrics so that they apply to graph drawings that do not just comprise straight-line edges but also polygonal or curved edges. For two drawings with different metric values showing the same (famous) graph, see Figure 1. This generalisation allows us to explore a wider range of graph drawings, and, arguably, a more representative set of graph drawings. The generalised metrics also allow us to investigate a set of graph drawings that we know are of real interest to the graph drawing community: the drawings in the proceedings of the Graph Drawing conference since 1998.

To this end, we used the Graph Drawing Harvester [5], which was originally designed to extract abstract graphs from a single PDF file (and to check whether the extracted graphs are already contained in the graph database *House of Graphs* [4]). We have extended the Harvester in three directions. Firstly, it can now run in batch, processing many articles one after the other, writing log files with the results. Second, it now checks more carefully for (false) multi-edges and loops. Finally, it now extracts, for each edge, a geometric description; that is, a polygonal line or a Bézier curve that connects its two endpoints. This allows us to apply the (generalised) metrics to the extracted drawings.



(a) Drawing with unit-length straight-line edges (AR 0.24, CA 0.8, EC 0.93, ELD 1, NP 0.09).

(b) Drawing with only two crossings (AR 0.33, CA 0.9, EC 0.97, ELD 0.72, NP 0.4).

Figure 1 Two drawings of the Petersen graph and a selection of their metric values (for the abbreviations and the definition of the metrics, see Section 3.2).

Our contribution. We first present a summary of the existing metrics, and explain how we have extended them to cater for drawings with curves or bends. The Graph Drawing Harvester pipeline and the challenges in its design and implementation are described next. We then present the collection of 4,890 graph drawings that we have extracted from 27 editions of the proceedings of the Symposium of Graph Drawing (and Network Visualization) 1998–2024. To validate this collection, the *GD-collection-v1*, we take a random sample of

extracted drawings from about 10% of the articles and analyse the quality of our extraction, in terms of true/false positives/negatives, and in terms of the number of missing or wrong features of the true positives. We present the metric landscape of the *GD-collection-v1* and compare it to the landscape of generated layouts defined by Mooney et al. [18].

Given the inevitable errors in the completely automated extraction, the *GD-collection-v1* is clearly not a perfect representation of the graph drawings contained in the proceedings. In spite of these errors, we think that this initial version is already a valuable benchmark set of mostly hand-drawn graph drawings that complements the existing benchmark sets (Rome graphs, AT&T graphs) that have been selected in a completely different way and that do not come with an "original" drawing. We will make the extraction code, the *GD-collection-v1*, and the metric code available to the community so that everybody can check the extracted drawings and add missing or remove erroneous features.

2 Background

2.1 Graph Drawing Metrics

To evaluate the aesthetic quality of a graph drawing, researchers define metrics that quantify the extent to which a graph drawing conforms to a particular layout principle. Typical layout principles include minimising edge crossings [21, 23, 24] or maximising symmetry [29].

These metrics are also used in layout algorithms to produce graph drawings by optimising different criteria. For example, Kamada and Kawai [12] explicitly optimise global distance preservation, while Radermacher et al. [24] minimise edge crossings. Graph drawing metrics are also used in human experiments to investigate the perception and task performance of graph layouts [19, 21, 28].

Purchase [22] defines a set of seven metrics, such as *Angular Resolution*, which are normalised to the range $[0, 1]$ to facilitate comparison across drawings of different graphs. Values of 1 represent the extreme that is intuitively assumed to be good, e.g., zero crossings. Mooney et al. [18] extend this set by refining existing metrics and introducing new ones, including the *Gabriel Ratio*. They present the distributions of metric values across different layouts and examine pairwise correlations among metrics. In this paper, we build upon their analysis by comparing these results to those from the *GD-collection-v1*.

The dataset of Mooney et al. [18] comprises 364,094 drawings generated by applying six layout algorithms (each five times) to a set of 16,768 graphs sourced from the North [9] and Rome [6] datasets, as well as from six random graph generators. The layout algorithms include: Fruchterman–Reingold [8], Kamada–Kawai [12], DRGraph [30], Sugiyama et al. [26], Human-like Orthogonal Network Layout (HOLA) [14], and a circular layout.

Due to some layout algorithms being deterministic and bugs in the HOLA implementation, not all combinations yielded unique or valid drawings, hence the dataset size falls short of the expected 503,040 (i.e., $16{,}768 \times 6 \times 5$). To address this, we extend the dataset by reapplying layout algorithms to node-reordered versions of the graphs, thereby generating additional distinct drawings. Despite encountering some further HOLA-related failures, our extended dataset includes 501,185 drawings. We call this the *extended Mooney dataset*.

2.2 Harvesting Graphs

While there are tools for recovering graph structures from either vector [5] or bitmap drawings [2], to the best of our knowledge, there is no tool that allows recovering complex edge geometries such as bends and Bézier curves. In the following, we focus on vector graphics because they are common for manual drawings and easier to process automatically.

Our graph harvesting pipeline consists of three major steps: figure detection, recognising vertex candidates, and edge path recovery. For figure detection, we rely on a modified version of the KIETA pipeline [13] that was originally built to detect tables. It uses keywords to recognise figure captions as well as clustering to find confined areas with many geometric objects. We use the rule-based approach of Deynet et al. [5] to recognise vertex candidates, but we slightly extended the ruleset. Finally, we implemented a greedy algorithm with another set of rules to extract the edge paths. In the process, we need to split each line segment or Bézier curve that contains a vertex in its relative interior.

3 Revised Graph Drawing Metrics

Existing metrics for evaluating graph drawings typically only apply to straight-line, connected, simple graphs. In practice, graph drawings often contain edges with bends/curves, may contain multiple edges between nodes, and comprise disconnected components. This section motivates and describes a new method for representing more complex graph drawings and discusses how the definitions and implementations of existing metrics must be altered to support this representation.

3.1 GEG Encodes Graphs

Existing graph storage formats (such as GraphML [27]) allow edge geometries to be stored as a collection of "bend" points, where these points describe a path of straight-line segments. Extensions to GraphML[1] and other niche formats may allow for edges to be stored as Bézier curves; however, these formats typically do not support (or are not widely used for) more complex edge geometries, which may comprise multiple straight-line and Bézier components. To extract drawings accurately, we must be able to faithfully represent complex edge geometries. We propose a JSON-based format for storing graph drawings: GEG (GEG Encodes Graphs).

GEG is essentially a list of nodes and edges, where each node has an "id" attribute, and each edge has a "source" and "target" attribute, which correspond to node ids. These are the minimum attributes required to describe a graph, but to store a graph drawing, node positions must also be defined. This can be done in two ways: define a "pos" attribute, which is a tuple representing the coordinates of the node, or define "x" and "y" attributes. While node positions are sufficient to encode straight-line drawings, we can also encode complex edge geometries in a "path" attribute, which is formatted as an SVG path command.[2]

This path format was chosen as SVG is a widely deployed format (i.e., with baseline features supported by all major browsers) which is capable of encoding the complex geometries we require. We expect each path to start with an "M" command, which "moves" to the source node's coordinates. This "M" command must be followed by one or more other SVG path commands, such as "L" for a straight-line segment, or "C" for a cubic Bézier curve, which eventually terminate at the coordinates of the target node. If no "path" attribute is included, the edge is assumed to be a straight-line.

A JSON schema for the GEG format and the code to parse GEG files and convert from other common graph formats is included with the metric implementations (see Section 6).

---

[1] `https://www.yworks.com/xml/schema/graphml/1.0/doc/http___www.yworks.com_xml_graphml/element/BezierEdge.html`

[2] Scalable Vector Graphics (SVG) 2 standard. `https://www.w3.org/TR/SVG/paths.html`

3.2 Metric Definitions

The following definitions extend those described by Mooney et al. [18]. We exclude their Gabriel Ratio metric, which is not applicable for drawings with curves, and we include a normalised metric for stress, which is not sensitive to scale and/or rotation [19].

Given a drawing D, let $G(D)$ be the graph that is represented by D, let $V(D)$ be the set of vertices of $G(D)$ (at the positions in D), let $E(D)$ be the set of edges of $G(D)$ (as drawn in D), let $X(D) \subseteq \mathbb{R}^2$ be the set of edge intersection points, and let $\mathcal{X}(D)$ be the set of pairs $(x, \{e, f\})$ that consist of an intersection point x and a pair $\{e, f\}$ of edges that cross at x. For an intersection point x, let $E(x)$ be the set of edges that cross at x.

Angular Resolution (AR) of a drawing D is the deviation of the smallest angle between two edges incident to a vertex compared to the ideal such angle of the same vertex, averaged over the vertices in $V_{>1}(D)$, the set of vertices of degree greater than 1:

$$\mathrm{AR}(D) = 1 - \frac{1}{|V_{>1}(D)|} \sum_{v \in V_{>1}(D)} \left| \frac{\vartheta_v - \vartheta_v^{\min}}{\vartheta_v} \right|, \tag{1}$$

where $\vartheta_v = 360° / \deg(v)$ is the ideal angle at vertex v and $\vartheta_v^{\min}$ is the actual minimum angle between any pair of edges incident to v.

Mooney et al. [18] used the same definition, but our implementation extends theirs; we account for self-loops and multi-edges, and we compute the angle between curved edges at a vertex using the tangent vectors defined by the control points of the Bézier curves.

Aspect Ratio (Asp) of a drawing D is the ratio between the height and width of the axis-aligned bounding box that encloses all vertices and edges. Formally, let $h(D)$ be the height and let $w(D)$ be the width of the bounding box of D. Then

$$\mathrm{Asp}(D) = \begin{cases} 1 & \text{if } h(D) = 0 \text{ or } w(D) = 0, \\ h(D)/w(D) & \text{if } h(D) \leq w(D), \\ w(D)/h(D) & \text{otherwise.} \end{cases}$$

Compared to Mooney et al. [18], our definitions of $h(D)$ and $w(D)$ now take the geometry of the edges into account.

Crossing Angle (CA) of a drawing D is the deviation of the crossing angles from 90°, averaged over every combination of crossing point x and pair of crossing edges crossing at x.

$$\mathrm{CA}(D) = 1 - \frac{1}{|\mathcal{X}(D)|} \sum_{(x,\{e,f\}) \in \mathcal{X}(D)} \left| \frac{\phi_{x,e,f} - \phi_{x,e,f}^{\min}}{\phi_{x,e,f}} \right|, \tag{2}$$

where $\phi_{x,e,f} = 90°$ is the ideal crossing angle of edges e and f at x and $\phi_{x,e,f}^{\min}$ is the smallest angle between edges e and f at x.

To calculate crossings, our implementation splits curved edges into straight-line segments, and checks for crossings between segment pairs. A parameter controls how many segments are used to estimate the curves, which we set to 100. We also include an angular tolerance parameter that specifies the minimum angle where a crossing should be counted (to prevent long overlapping edges being counted as many crossings). We set this to 2.5°. This approach also applies to *Edge Crossings*.

Edge Crossings (EC) of a drawing D counts the number of edge crossings in D, scaled against the total number of potential crossings (in a straight-line drawing of $G(D)$) [22]:

$$\mathrm{EC}(D) = 1 - \begin{cases} c/c_{\max} & \text{if } c_{\max} > 0, \\ 1 & \text{if } c > c_{\max}, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where $c = \sum_{x \in X(D)} |E(x)|^2$ is the number of crossings in D, $c_{\max} = \binom{|E(G)|}{2} - c_{\deg}$ is an upper bound on the number of possible crossings (in a straight-line drawing), and $c_{\deg} = \sum_{v \in V(D)} \binom{\deg(v)}{2}$ is the number of crossings that are impossible due to the fact that adjacent edges cannot cross. A drawing that admits polygonal or curved edges can have more than $c_{\max}$ crossings (as, for example, the assumption that neighbouring edges cannot cross no longer holds). For such a drawing D, we set $\mathrm{EC}(D) = 0$.

Edge Length Deviation (ELD) of a drawing D is the average relative deviation of edge lengths from an ideal value [1]:

$$\mathrm{ELD}(D) = 1 \Bigg/ \left(1 + \frac{1}{|E(D)|} \sum_{e \in E(D)} \left| \frac{L(e) - L_{\mathrm{ideal}}(D)}{L_{\mathrm{ideal}}(D)} \right| \right), \tag{4}$$

where $L(e)$ is the geometric length of edge e, either as the arc length of a curved path or the Euclidean length of a straight-line segment, and $L_{\mathrm{ideal}}(D) = (\sum_{e \in E(D)} L(e))/|E(D)|$ is the average edge length of D. $\mathrm{ELD}(D) = 1$ if and only if all edges of D have the same length.

We refine the metric by explicitly specifying the reciprocal normalisation, thereby improving clarity and interpretability of this metric.

Edge Orthogonality (EO) of a drawing D measures how closely the direction of each edge aligns with the horizontal or vertical axes. For each edge, we compute the angular deviation of its segments from the nearest orthogonal direction (0°, 90°, or 180°), scaled so that 0 indicates perfect alignment and 1 indicates a 45° diagonal. This definition extends that of Mooney et al. [18] to support the inclusion of curved edges.

$$\mathrm{EO}(D) = 1 - \frac{1}{|E(D)|} \sum_{e \in E(D)} \delta_e, \text{ where} \tag{5}$$

$$\delta_e = \sum_{j=1}^{k_e} \frac{\min(\theta_{e,j}, |90° - \theta_{e,j}|, 180° - \theta_{e,j})}{45°} \cdot \frac{\ell_{e,j}}{\tilde{L}(e)} \tag{6}$$

is the length-weighted orthogonality deviation of edge e, each edge e is approximated by a polygonal line $P(e)$ with k_e straight-line segments, $\theta_{e,j}$ is the absolute angle between segment j of $P(e)$ and the horizontal axis, $\ell_{e,j}$ is the length of segment j of $P(e)$, and $\tilde{L}(e) = \sum_{j=1}^{k_e} \ell_{e,j}$ is the total length of $P(e)$.

Kruskal Stress Metric (KSM) is a quality metric adapted from classical multidimensional scaling, originally proposed by Kruskal [15]. It quantifies how well the *relative ordering* of graph-theoretic distances is preserved in the layout, rather than requiring exact spatial accuracy. A more detailed description is given in [25].

To compute the metric, a *Shepard diagram* is constructed by plotting each pairwise Euclidean distance in the layout against the corresponding graph-theoretic (shortest-path) distance. An isotonic regression is then fitted to these points to model the ideal monotonic relationship. The Kruskal Stress Metric of a drawing D is defined as:

$$\mathrm{KSM}(D) = 1 - \sqrt{\frac{\sum_{u,v \in V(D),\, u \neq v} \left(\|u - v\| - \hat{d}_{u,v}\right)^2}{\sum_{u,v \in V(D),\, u \neq v} \|u - v\|^2}} \tag{7}$$

where $\hat{d}_{u,v}$ is the fitted distance from an isotonic regression applied to the Shepard diagram. To maintain consistency with other metrics, we subtract the stress value from 1 (which makes a lower stress correspond to a higher score).

Neighbourhood Preservation (NP) of a drawing D measures how well graph distances (in $G(D)$) between vertices are reflected by their Euclidean distances in D. This is expressed by the Jaccard similarity between the adjacency matrix of $G(D)$ and a geometric k-neighbourhood matrix [1]. Formally,

$$\mathrm{NP}(D) = \frac{|A \wedge M^k|}{|A \vee M^k|}, \tag{8}$$

where A is the adjacency matrix of $G(D)$ (i.e., $A_{uv} = 1$ if vertices u and v are adjacent) and M^k is a binary matrix such that $M^k_{uv} = 1$ if and only if vertex v is one of the k nearest Euclidean neighbours of vertex u in D. We set $k = \lfloor 2|E(D)| \,/\, |V(D)| \rfloor$ to the floor of the average degree.

Node Resolution (NR) of a drawing D is the ratio of the minimum to the maximum Euclidean distances over all pairs of vertices in D. Formally,

$$\mathrm{NR}(D) = \frac{\min_{u,v \in V(D),\, u \neq v} \|u - v\|}{\max_{u,v \in V(D),\, u \neq v} \|u - v\|}. \tag{9}$$

Node Uniformity (NU) of a drawing D measures how evenly vertices are spatially distributed across D. To this end, the bounding box of D is divided into a uniform grid with T (to be determined) rectangular cells, and the deviation between the actual and the ideal number of vertices per cell is used to compute the score. Let n_i denote the number of vertices in cell i, and let $\mu = |V(D)|/T$ be the ideal number of nodes per cell. Then:

$$\mathrm{NU}(D) = 1 - \frac{1}{D_{\max}} \sum_{i=1}^{T} |c_i - \mu|, \tag{10}$$

where $D_{\max} = 2|V(D)|(T-1)/T$ is the worst-case deviation when all nodes are concentrated in a single cell. We set T to match $|V(D)|$ while maintaining roughly square proportions. Specifically, we set the grid dimensions to

$$\#\mathrm{rows} = \left\lfloor \sqrt{|V(D)|} \right\rfloor, \quad \#\mathrm{cols} = \left\lceil \frac{|V(D)|}{\#\mathrm{rows}} \right\rceil, \quad \text{and, hence,} \quad T = \#\mathrm{rows} \cdot \#\mathrm{cols}.$$

If the bounding box of D has height (or width) 0, then $\#\mathrm{rows} = 1$ (or $\#\mathrm{cols} = 1$).

This definition refines the metric proposed by Mooney et al. [18] by improving the handling of cases where the number of vertices is not a square number. Our approach determines the grid dimensions to maintain near-square proportions, while matching the vertex number as closely as possible. Additionally, the normalisation is now based on a well-defined worst-case scenario (all vertices contained in a single cell) rather than relying on an ad-hoc damping mechanism that adjusts scores outside the range $[0, 1]$.

3.3 Directed and Disconnected Graphs

One limitation of these metrics is that they do not consider directed graphs. While they can be applied by converting to undirected graphs (with multi-edges if needed), some layout principles may not translate directly – for example, smaller angles between edges (angular resolution) may be preferable to emphasise directionality.

These metrics can generally be applied to disconnected graphs. For the two that definitionally require connectedness (Kruskal Stress Metric, Neighbourhood Preservation), we apply them to each connected component and return a weighted sum, where weights are based on each component's convex hull area relative to the total.

While the weighted sum approach could be applied to all metrics, it is not always necessary or appropriate. For example, Angular Resolution is locally defined at each node and unaffected by disconnected components. In contrast, using a weighted sum for Edge Crossings would be misleading, as it would ignore crossings between components – an issue if disconnected parts overlap in the drawing. Although such overlap should be avoided to prevent misinterpreting the graph as connected, it may still occur in practice.

4 Extended Graph Harvester

We use and extend Graph Harvester [5], an automated pipeline for extracting vector graph drawings from PDFs. We applied it to the Graph Drawing conference proceedings from 1998–2024. Proceedings from 1997 and earlier are only available as scanned documents and therefore inaccessible to our pipeline that can only handle vector graphics.

4.1 Extraction Logic

The process of harvesting graph drawings can be described by three major phases, namely figure detection, recognising vertex candidates, and edge path recovery. Graph Harvester focuses on the most prevalent type of graph drawings, node-link-diagrams. Here, vertices are represented as individual geometric objects (e.g., small disks) and edges are represented by connecting nodes by straight-lines or curves. We use the term node to refer to the geometric representation of a vertex.

PDFs usually lack semantic information about document structure, so figures must be detected by identifying distinctive patterns (e.g., "Figure" followed by a number and some delimiter) and by analysing areas with many geometric objects and irregular text arrangements. A single figure can contain multiple graph drawings, which can be distinguished by partitioning the figure into non-overlapping clusters of elements.

Naturally, the second step is finding vertex candidates. Usually, all vertices are represented by objects of the same shape and size, but some vertices may use different or larger shapes in order to highlight the vertex or label certain features. Figures also often contain other geometry for annotations that can be mistaken for vertices.

In a third step, we handle edge paths. Ideally, nodes are connected by a distinct sequence of stroked shapes (line segment or Bézier curves) that each start or end at either a node or endpoint of the previous stroked shape in the sequence. This assumption falls short of several patterns used by authors of graph drawings. When drawing self loops, the start and end vertex are identical. Multiedges connect the same pair of nodes via different paths. One entity of geometry may be used to draw multiple edges (e.g., a rectangle that connects four nodes on its corners and therefore represents four edges or a long line segment that

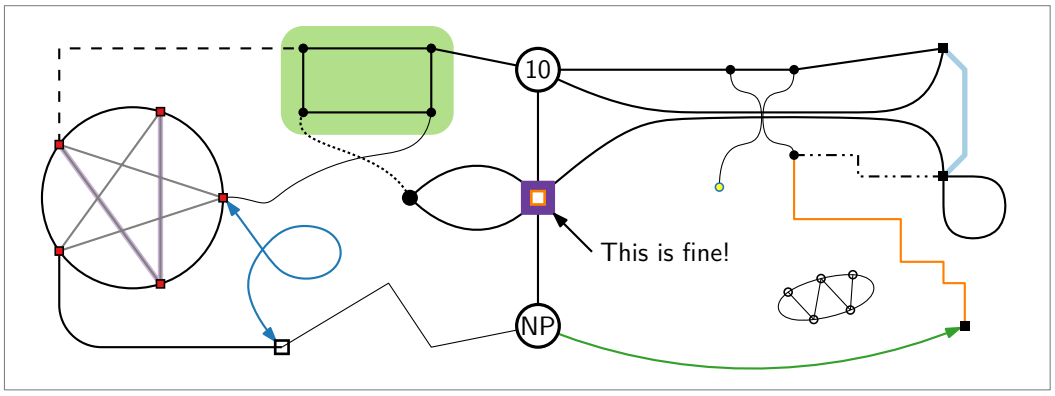


Figure 2 A "test chart" for what the enhanced Graph Harvester can extract. It recognizes all nodes and edge paths; the text labels are filtered out automatically.

connects multiple collinear nodes). Common annotations on edges include arrow heads for expressing edge direction and multi patterning where a thicker shape is placed under the edge for highlighting purposes.

For the Graph Harvester we rely on the following assumptions. Nodes are either squares or circles, they are either stroked (i.e., only the outline is drawn) or filled using a dark colour, and they are clearly separated from each other. Fill may be implemented as multiple stacked shapes of different size (see for example the centre node of Figure 2).

Edges are non-empty sequences of line segments or Bézier curves without jumps (i.e., every segment connects to the next at their respective end/start point), they have a dark stroke, and they connect exactly two nodes (that may be the same node). We recognise edge paths by first splitting any geometry that intersects a vertex candidate, then, matching every endpoint to any overlapping vertex candidate and finally, greedily extending unmatched endpoints with other unmatched pieces of geometry. When extending edge paths, we forbid very acute angles (smaller than 60°) as edges tend to extend towards their destination.

Any remaining geometry (i.e., any path that has an unmatched endpoint) is discarded as annotation as well as all sort of text. Short self loops are also discarded as they usually result from arrow heads or node decorations. We ignore some properties of the geometry such as colour and dash patterns as they are used in many different ways and often highlight vital parts of the drawing. However, we remove very lightly coloured geometry (Rec. 709 luma ≥ 0.75) as these are mainly used for annotation or are not visible at all. After reviewing first results (see Section 4.2), we decided to exclude all drawings where self-loops were detected because true self-loops are very rare in the dataset and the majority of detections stemmed from completely unusable extractions. We also excluded some very large graph drawings for performance reasons.

We composed a graph drawing using a diverse set of styles that match our assumptions, similar to a test chart for imaging systems (see Figure 2). Our extended version of Graph Harvester handles this correctly.

4.2 Evaluation

In order to asses the quality of our graph extraction pipeline, we manually examined 101 papers (about 10%) that we picked at random from our dataset. In these we found 1,358 figures, 1,016 of which were graph drawings (i.e., node-link-diagrams). Note that we counted clearly separated drawings individually even when they were composed into a single figure (e.g., using subfigures). See Figure 3 for an overview.

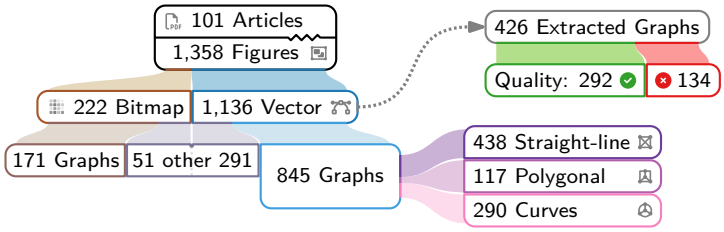


Figure 3 We analysed a sample of 101 articles manually in order to validate our large corpus.

So far, Graph Harvester cannot detect if an edge is directed or not. In our sample, 12% of the graphs were directed (which Graph Harvester can't detect), 11% were multi-graphs (i.e., had multiple edges connecting the same pair of nodes), and 1% had self loops. When looking at the actual drawings, 54% used only straight-line edges, 12% contained at least some polygonal edges, and 34% made use of curved edges. Because our pipeline can only handle vector graphics, we loose all bitmap images in the extraction process. Fortunately, only 17% of the graph drawings were bitmaps.

From the sample, Graph Harvester managed to extract 426 graphs. In 26 cases multiple proximal drawings were detected as one disconnected graph. So the extracted graphs actually stem from 487 individual drawings. For numerous reasons (see Section 4.3 for some examples), 365 graphs could not be extracted. Only seven other figures were incorrectly detected as graphs. For the extracted graphs, we checked how accurate they reproduce the adjacencies and edge paths depicted. An extraction "passes" our assessment if all vertices are recognised and only an insignificant number of edges is missing or misrepresented. In our sample, 69% of the extracted graphs passed resulting in an overall somewhat low rate of 40% high-quality extractions from the vector graph drawings.

Given the very diverse landscape of drawing styles and our generous interpretation of what to count as a graph drawing (see Section 4.3), we still consider these numbers a success. We argue that the quality of our extraction procedure is sufficient to make further assessments based on this dataset.

4.3 Limitations

To explore limitations of our approach, we focus on those graph drawings that caused problems in the extraction process. We compiled the following list of common patterns. Each item refers to the corresponding tile in Figure 4 (a) to (o).

(a) Not a Node-Link-Diagram (but a graph drawing). The figure shows a so-called contact representation. We consider graph drawings that are not node-link-diagrams out of scope (i.e., we did not count them in Section 4.2 and do not expect any extracted graphs).

(b) Not a Graph Drawing (but a node-link-diagram). Some figures that are not intended to show graphs contain geometry that can be mistaken for nodes and edges. We use a set of filter rules in order to prevent such extractions. Few false positives were reported in the evaluation.

(c) Incomplete Drawing. The complete parts are extracted; incomplete edges are discarded.

(d) Disconnected Components and Isolated Vertices. If the drawing is detected as one figure, a disconnected graph is extracted. All isolated vertices are discarded.

(e) Huge and/or Heavily Annotated Nodes. These node types are usually not identified as vertex candidates, so false detections often occurred with such drawings.

(f) Text-only Nodes. No graph is extracted.

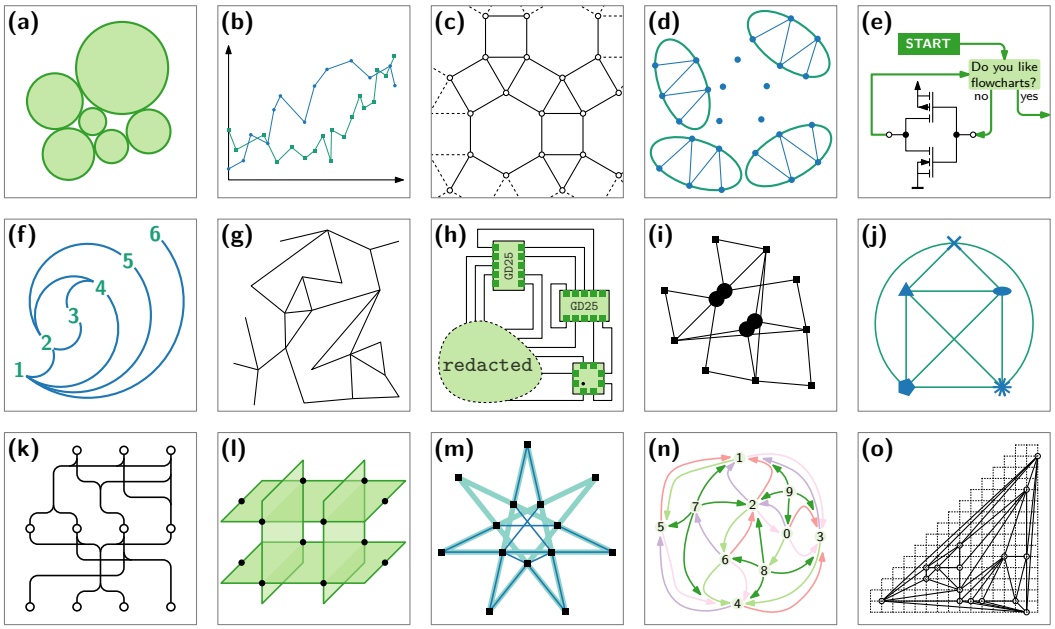


Figure 4 Things we actually found in graph drawings.

(g) No Vertex Markers. No graph is extracted.

(h) Ports or Compose Nodes and Intentionally Obstructed Elements. Nodes of this style are usually not identified as vertex candidates. If geometry is present in the vector data, it is extracted regardless of any patch occluding it.

(i) Partially Overlapping and/or Ambiguously Placed Nodes Partially overlapping nodes are filtered and only one of them is extracted. If a node overlaps an edge, that edge is split into two edges adjacent to that node.

(j) Unusual Node Shapes (i.e., not a square or a circle). No graph is extracted.

(k) Confluent Edges and Hyper Edges (i.e., edges that connect more than two nodes). If each confluent edge consists of an individual geometry object, the extraction succeeds. Otherwise, at most one edge per hyper edge is extracted.

(l) Acute Edge Bends and Filled Edges. Segments resulting in acute angles are not considered for edge continuations. Filled geometry is considered as edge paths.

(m) Two-in-one Drawing. A blend of both graphs is extracted. Multiply exposed edges are filtered and only one copy is extracted.

(n) Lightly Coloured Elements and Edges not Reaching Their Target. Very lightly coloured elements are discarded. Incomplete edge paths are stretched only to a certain degree to prevent false extractions.

(o) Grid or Level Lines. Unfortunately, these are extracted as edge paths.

5 Graph Drawings in the GD Proceedings

5.1 General Analysis

From the 27 proceedings (1998–2024) we extracted 1,037 articles. Of these articles, only 740 (71.4%) contain graph drawings (that we were able to extract). In total we extracted 4,890 graph drawings – 3,769 (77.1%) were connected, 858 (17.6%) were multi-graphs, and 4,211 (86.1%) were planar (though only 2,807 (57.4%) were actually drawn without crossings).

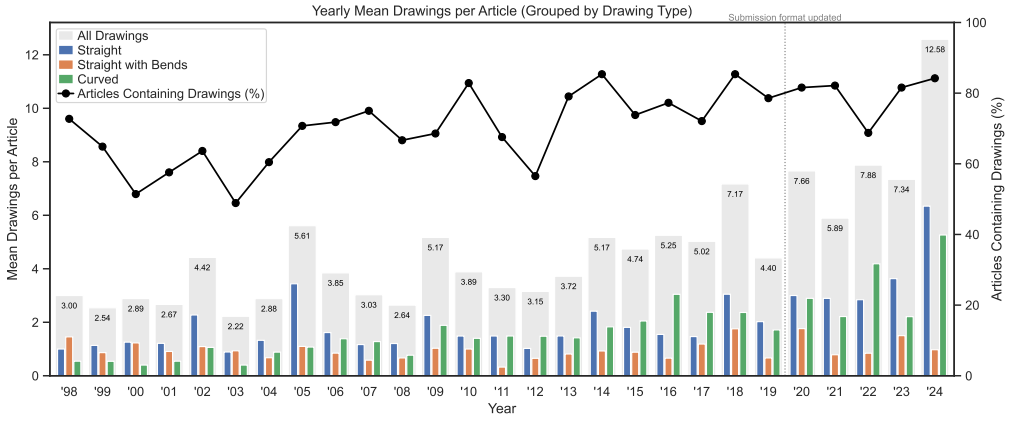


Figure 5 The stacked bar chart shows, for each year, the mean number of drawings per article (grouped by edge style). The line chart shows, for each year, the proportion of articles that contain drawings. In 2020, the submission format for GD was updated to support the inclusion of figures.

We categorise the 4,890 drawings into three sets ("styles") based on edge geometry:

- **Straight**: Those which consist of only straight-line edges: 2,102 (43.0%)
- **Polygonal**: Those which contain polygonal edges (i.e., bends): 998 (20.4%)
- **Curved**: Those which contain curved edges: 1,790 (36.6%)

The significant proportion of drawings which contain curves highlight the importance of defining metrics that can evaluate such drawings.

Figure 5 shows the mean number of drawings per article per year, grouped by drawing type. We fitted linear regressions to examine trends in the number of drawings per article over time. The mean number of drawings per article has increased significantly (slope = 0.22, $R^2 = 0.59$, $p < .001$), as has the proportion of articles that contain drawings (slope = 0.0088, $R^2 = 0.46$, $p < .001$). While these regressions confirm a clear upward trend over the past 27 years, this trend is not expected to continue indefinitely, as there are practical limits to how many drawings can be included in an article. The 2020 change from page limits to line limits – intended to allow more figures – may also have contributed to the observed trend, although our dataset only includes extracted graph drawings (not all figures, or even all drawings). Year-to-year differences also remain substantial, likely due to variations in conference themes, paper content, and editorial guidelines.

Figure 6 shows the distribution of drawings per article, excluding articles with no (extractable) drawings. The median article contains 5 drawings (mean = 6.61). The article with the most drawings (namely 67) was a 2009 paper by J. Joseph Fowler [7] – it contained many small planar trees.

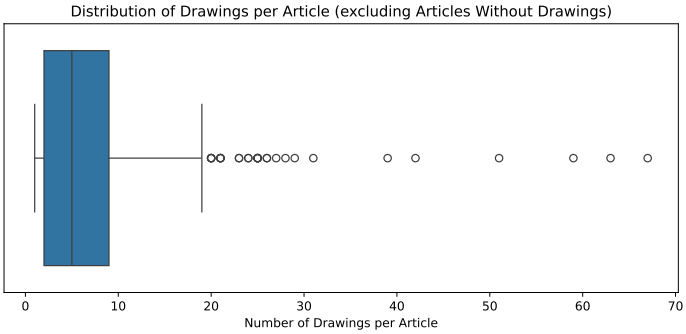


Figure 6 Boxplot of the number of drawings per article, excluding articles with no (extractable) drawings. The median article contains 5 drawings.

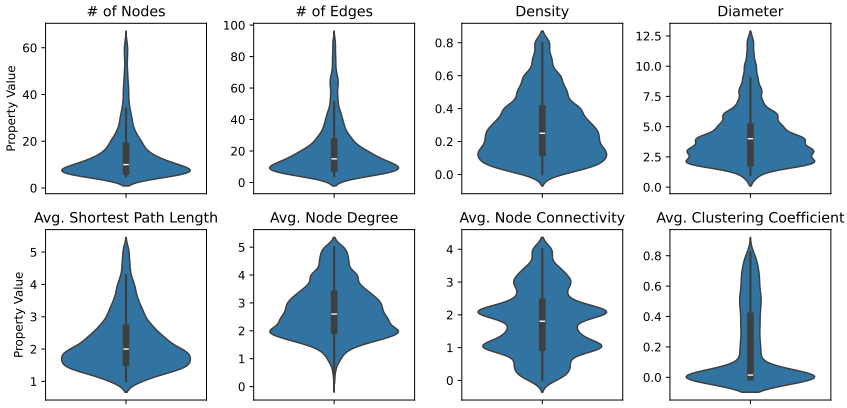


Figure 7 Violin plots showing the distribution of values for 8 common graph properties on the *GD-collection-v1*. Values above the 95th percentile have been excluded to allow for easier inspection of long tail-distributions (such as the number of nodes or edges).

Figure 7 shows the distribution of values for eight common graph properties on our dataset, excluding those above the 95th percentile to allow for easier comparison of long-tail distributions. The complete distributions are shown in Appendix A. All properties were calculated using implementations from the Python library NetworkX [10]. The average node connectivity, defined by Beineke et al. [3], measures the mean number of node-independent paths between all node pairs, reflecting overall network cohesion. The average clustering coefficient, defined by Kaiser [11], captures the tendency of nodes to form local clusters by averaging the proportion of links between each node's neighbours. The density of a graph G is calculated using the standard definition: $d(G) = |E(G)|/\binom{|V(G)|}{2}$. For graphs, density is in $[0, 1]$, but the density of multigraphs may exceed 1.

The majority of extracted graphs have very few nodes (median = 11), and are generally sparse (median density = 0.27). These factors contribute to low diameters (median = 4), average shortest path lengths (median = 2), and average node degrees (median = 2.67). The graphs also tend to have low average node connectivity (median = 1.98), reflecting limited redundancy in paths between node pairs. Average clustering coefficients are similarly low (median = 0.06), suggesting limited local grouping or community structure.

5.2 Metric Analysis of Drawings

In this section we analyse our dataset using the ten readability metrics defined in Section 3.2. Figure 8 shows the value distributions of the ten metrics across our 4,890 extracted drawings (separated by edge type) and the *extended Mooney dataset* (see Section 2.1). Table 1 shows the medians for each metric and dataset (other statistics available in Appendix C).

First, we compare all drawings in the *GD-collection-v1* to the *extended Mooney dataset*. Overall the medians for all metrics are lower in the *extended Mooney dataset* than the *GD-collection-v1*. We attribute this to the greater concentration of larger graphs – the median number of nodes in the *GD-collection-v1* is 11, compared to 48 in the *extended Mooney dataset*. Larger graphs are expected to have poorer performance on most metrics. Full distributions of graph properties for both datasets are provided in Appendix A.

Comparing the three edge geometry types in the *GD-collection-v1* we can also see that distributions are largely similar, though differences highlight some interesting properties. The *Crossing Angle* metric shows a larger distribution of values around 0.7 for the curved drawings when compared to the straight/polygonal drawings. We suggest this stems from arc diagrams and other similar drawings where crossing angle is not optimised.

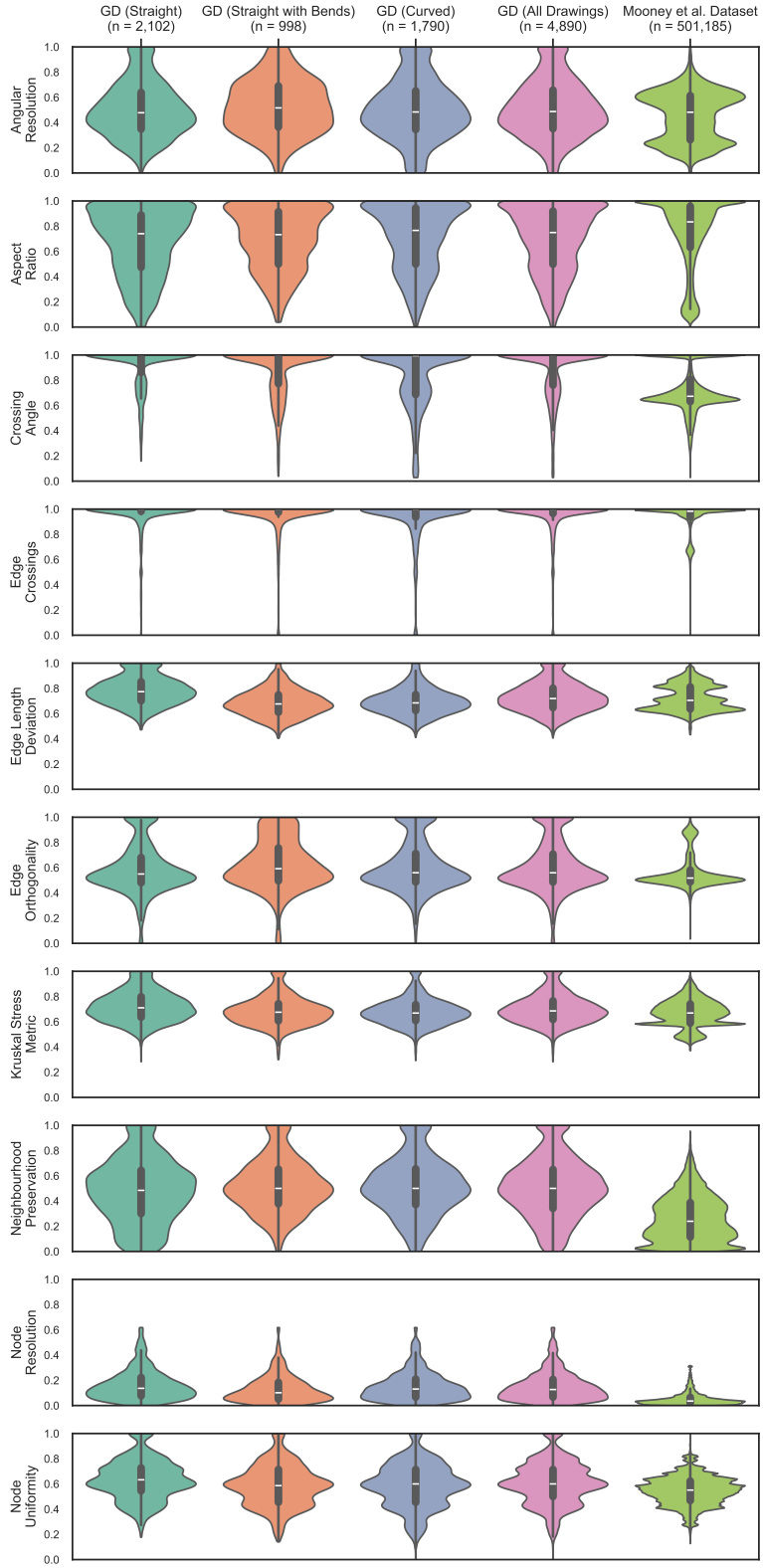


Figure 8 Violin plots showing the distribution of values for 10 metrics on the *GD-collection-v1* (separated by edge type) and the *extended Mooney dataset*.

Table 1 Median for each metric and dataset (see Table 2 in the Appendix for quartiles).

Metric	Straight	Polygonal	Curved	GD Collection v1	Mooney
Angular Resolution	0.478	0.516	0.484	0.488	0.482
Aspect Ratio	0.74	0.733	0.766	0.75	0.834
Crossing Angle	1.0	1.0	0.995	1.0	0.673
Edge Crossings	1.0	1.0	0.998	1.0	0.983
Edge Length Deviation	0.774	0.676	0.685	0.719	0.705
Edge Orthogonality	0.55	0.592	0.56	0.559	0.517
Kruskal Stress Metric	0.71	0.675	0.669	0.686	0.669
Neighbourhood Preservation	0.484	0.5	0.5	0.5	0.239
Node Resolution	0.136	0.101	0.131	0.127	0.036
Node Uniformity	0.633	0.586	0.6	0.6	0.551

Overall, the drawings have very few crossings, but curved drawings tend to admit more crossings than the other two sets (median = 0.999 vs. 1.0). Straight-line drawings clearly perform better on the *Edge Length Deviation* metric, reflecting more consistent edge lengths compared to the variability introduced by curves or bends (median = 0.772 vs. polygonal: 0.677; curved: 0.685). Polygonal-edge drawings perform better for the *Edge Orthogonality* metric (median = 0.589 vs. straight: 0.552; curved: 0.56). We suggest this may be caused by orthogonal layouts which typically contain straight edges with bend points at 90°.

Straight-line drawings perform better on the *Kruskal Stress Metric* (median = 0.709 vs. polygonal: 0.676; curved: 0.669), likely because many stress-minimising layout algorithms are tailored to straight-line representations, even though the metric itself depends only on node positions. Straight-line drawings also perform better on the *Node Uniformity* metric (median = 0.625 vs. polygonal: 0.586; curved: 0.6). Straight-line layouts typically position nodes to reflect structural relationships in the graph, resulting in more uniform spatial distributions. In contrast, bent or curved edge layouts often prioritise edge clarity or visual grouping, which can lead to clustered or uneven node placements and consequently lower node uniformity.

We also examined the pairwise correlations between metrics across both datasets. Overall, the two datasets exhibit generally consistent correlation directions between metric pairs; however, correlation strengths vary. A few discrepancies exist, which we attribute to differences in the types and structures of graphs and layouts included in each dataset. The correlation heatmaps are provided in Appendix B.

6 Conclusion

In this paper, we revised a set of ten quality metrics for evaluating graph drawings, generalising them beyond the traditional constraint of straight-line edges to include drawings with bends and curves.[3] We extended the *Graph Harvester* [5] to obtain an automated pipeline for extracting node-link diagrams – including edge geometries – from vector-based figures in PDFs. Using this tool, we created the *GD-collection-v1*, a dataset of 4,890 graph drawings extracted from 27 years of Graph Drawing conference proceedings.[4] Finally, we analysed this collection and compared its metric landscape to that of an extended benchmark set of over half a million drawings described by Mooney et al. [18].

[3] The implementation of the ten metrics and GEG format is available at `https://github.com/gavjmooney/geg`

[4] The full dataset is available at `https://github.com/hegetim/gd-collection`

Our analysis suggests that while existing readability metrics are broadly applicable, applying them to real-world graph drawings, such as those in the *GD-collection-v1*, reveals subtle differences not evident in datasets comprised solely of synthetically generated layouts. Differences in metric distributions across straight-line, polygonal, and curved drawings indicate that certain drawing styles tend to score better on different graph drawing metrics. The high proportion of curves (37%) and bends (20%) in the *GD-collection-v1* highlights the importance of metric definitions that are applicable to such drawings.

We plan to further grow the GD collection in the future. The community can support this by adhering to some basic guidelines when it comes to graph drawings: Do use vertex markers, use a single shape per edge, and avoid annotations that can be mistaken for edges. In future, more of the existing metrics defined only on straight-line drawings could be extended to work on curved drawings (e.g. those implemented in [20]).

References

1 Reyan Ahmed, Felice De Luca, Sabin Devkota, Stephen G Kobourov, and Mingwei Li. Multicriteria scalable graph drawing via stochastic gradient descent, (SGD)^2. *IEEE Trans. Vis. Comput. Graphics*, 28(6):2388–2399, 2022. doi:10.1109/TVCG.2022.3155564.

2 Christopher Auer, Christian Bachmaier, Franz J. Brandenburg, Andreas Gleißner, and Josef Reislhuber. Optical graph recognition. *J. Graph Algorithms Appl.*, 17(4):541–565, 2013. doi:10.7155/JGAA.00303.

3 Lowell W. Beineke, Ortrud R. Oellermann, and Raymond E. Pippert. The average connectivity of a graph. *Discrete Math.*, 252(1):31–45, 2002. doi:10.1016/S0012-365X(01)00180-7.

4 Kris Coolsaet, Sven D'hondt, and Jan Goedgebeur. House of Graphs 2.0: A database of interesting graphs and more. *Discret. Appl. Math.*, 325:97–107, 2023. doi:10.1016/J.DAM.2022.10.013.

5 Julius Deynet, Tim Hegemann, Sebastian Kempf, and Alexander Wolff. Graph Harvester. In Stefan Felsner and Karsten Klein, editors, *Graph Drawing and Network Visualization (GD)*, volume 320 of *LIPIcs*, pages 58:1–58:3. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. Poster with software presentation. doi:10.4230/LIPIcs.GD.2024.58.

6 Giuseppe Di Battista, Ashim Garg, Giuseppe Liotta, Roberto Tamassia, Emanuele Tassinari, and Francesco Vargiu. An experimental comparison of four graph drawing algorithms. *Comput. Geom.*, 7(5):303–325, 1997. doi:10.1016/S0925-7721(96)00005-3.

7 J. Joseph Fowler. Characterization of unlabeled radial level planar graphs. In David Eppstein and Emden R. Gansner, editors, *Graph Drawing (GD)*, volume 5849 of *LNCS*, pages 81–93. Springer, 2010. doi:10.1007/978-3-642-11805-0_10.

8 Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Softw. Pract. Exper.*, 21(11):1129–1164, 1991. doi:10.1002/spe.4380211102.

9 Emden Gansner, Eleftherios Koutsofios, and Stephen North. Drawing graphs with dot. Technical report, AT&T Research, 2006. URL: https://archive.org/details/dotguide.

10 Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Python in Science Conference (SciPy)*, pages 11–15, 2008.

11 Marcus Kaiser. Mean clustering coefficients: The role of isolated nodes and leafs on clustering measures for small-world networks. *New J. Phys.*, 10(8):083042, 2008. doi:10.1088/1367-2630/10/8/083042.

12 Tomihisa Kamada and Satoru Kawai. An algorithm for drawing general undirected graphs. *Inf. Process. Lett.*, 31(1):7–15, 1989. doi:10.1016/0020-0190(89)90102-6.

13 Sebastian Kempf, Markus Krug, and Frank Puppe. KIETA: key-insight extraction from scientific tables. *Appl. Intell.*, 53(8):9513–9530, 2023. doi:10.1007/S10489-022-03957-8.

14 Steve Kieffer, Tim Dwyer, Kim Marriott, and Michael Wybrow. HOLA: Human-like Orthogonal Network Layout. *IEEE Trans Vis. Comput. Graphics*, 22(1):349–358, 2016. doi:10.1109/TVCG.2015.2467451.

15 Joseph B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27, 1964. doi:10.1007/BF02289565.

16 Gavin J. Mooney, Tim Hegemann, Alexander Wolff, Michael Wybrow, and Helen C. Purchase. GD-collection-v1. Dataset, swhId: swh:1:dir:478a27dd277dc5818bdf699d2a5bc222a010533b (visited on 2025-11-10). URL: https://github.com/hegetim/gd-collection, doi:10.4230/artifacts.25065.

17 Gavin J. Mooney, Tim Hegemann, Alexander Wolff, Michael Wybrow, and Helen C. Purchase. GEG Encodes Graphs. Software, swhId: swh:1:dir:91f45ae7976a74b00a0bf86145b52dd78838fb29 (visited on 2025-11-10). URL: https://github.com/gavjmooney/geg, doi:10.4230/artifacts.25066.

18 Gavin J. Mooney, Helen C. Purchase, Michael Wybrow, and Stephen G. Kobourov. The multi-dimensional landscape of graph drawing metrics. In *IEEE Pacific Vis. Conf. (PacificVis)*, pages 122–131, 2024. doi:10.1109/PACIFICVIS60374.2024.00022.

19 Gavin J. Mooney, Helen C. Purchase, Michael Wybrow, Stephen G. Kobourov, and Jacob Miller. The perception of stress in graph drawings. In Stefan Felsner and Karsten Klein, editors, *Graph Drawing and Network Visualization (GD)*, volume 320 of *LIPIcs*, pages 21:1–21:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.GD.2024.21.

20 Martin Nöllenburg, Sebastian Röder, and Markus Wallinger. GdMetriX – A NetworkX extension for graph drawing metrics. In Stefan Felsner and Karsten Klein, editors, *Graph Drawing & Network Vis. (GD)*, volume 320 of *LIPIcs*, pages 45:1–45:3. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.GD.2024.45.

21 Helen C. Purchase. Which aesthetic has the greatest effect on human understanding? In Giuseppe Di Battista, editor, *Graph Drawing*, volume 1353 of *LNCS*, pages 248–261. Springer, 1997. doi:10.1007/3-540-63938-1_67.

22 Helen C. Purchase. Metrics for graph drawing aesthetics. *J. Vis. Lang. Comput.*, 13(5):501–516, 2002. doi:10.1006/jvlc.2002.0232.

23 Helen C. Purchase, Robert F. Cohen, and Murray James. Validating graph drawing aesthetics. In G. Goos, J. Hartmanis, J. van Leeuwen, and Franz J. Brandenburg, editors, *Graph Drawing*, volume 1027 of *LNCS*, pages 435–446. Springer, 1996. doi:10.1007/BFb0021827.

24 Marcel Radermacher, Klara Reichard, Ignaz Rutter, and Dorothea Wagner. A geometric heuristic for rectilinear crossing minimization. In *Workshop Algorithm Engin. & Exper. (ALENEX)*, pages 129–138. SIAM, 2018. doi:10.1145/3325861.

25 Kiran Smelser, Jacob Miller, and Stephen Kobourov. "Normalized stress" is not normalized: How to interpret stress correctly. ArXiv preprint, 2024. doi:10.48550/arXiv.2408.07724.

26 Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Syst. Man Cybern.*, 11:109–125, 1981. doi:10.1109/TSMC.1981.4308636.

27 GraphML Team. The graphml file format, 2002. URL: http://graphml.graphdrawing.org/.

28 Colin Ware, Helen Purchase, Linda Colpoys, and Matthew McGill. Cognitive Measurements of Graph Aesthetics. *Inform. Vis.*, 1(2):103–110, 2002. doi:10.1057/palgrave.ivs.9500013.

29 Erik Welch and Stephen Kobourov. Measuring symmetry in drawings of graphs. *Comput. Graph. Forum*, 36(3):341–351, 2017. doi:10.1111/cgf.13192.

30 Minfeng Zhu, Wei Chen, Yuanzhe Hu, Yuxuan Hou, Liangjun Liu, and Kaiyuan Zhang. DRGraph: An efficient graph layout algorithm for large-scale graphs by dimensionality reduction. *IEEE Trans. Vis. Comput. Graphics*, 27(2):1666–1676, 2021. doi:10.1109/TVCG.2020.3030447.

A Complete Graph Property Distributions

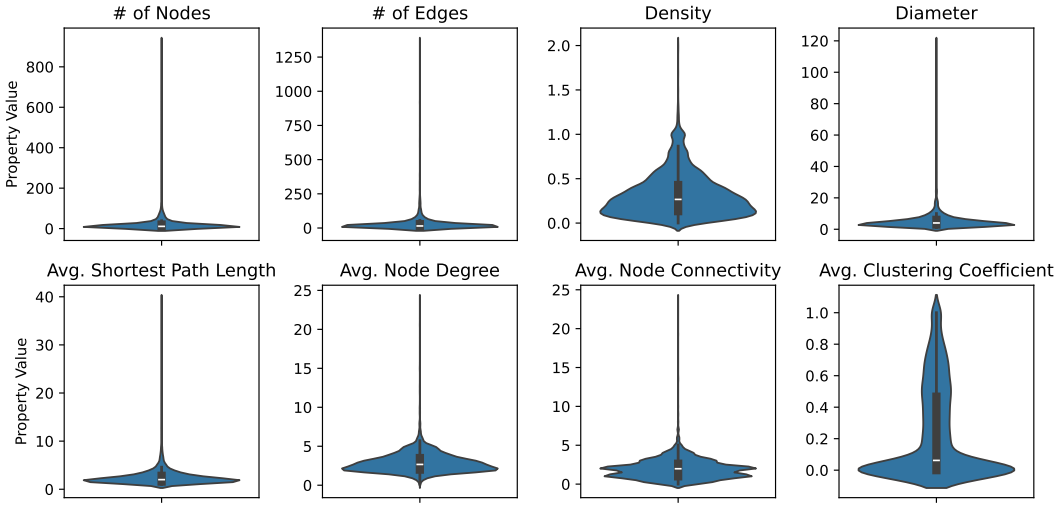


Figure 9 Violin plots showing the distribution of values for 8 common graph properties on the *GD-collection-v1*.

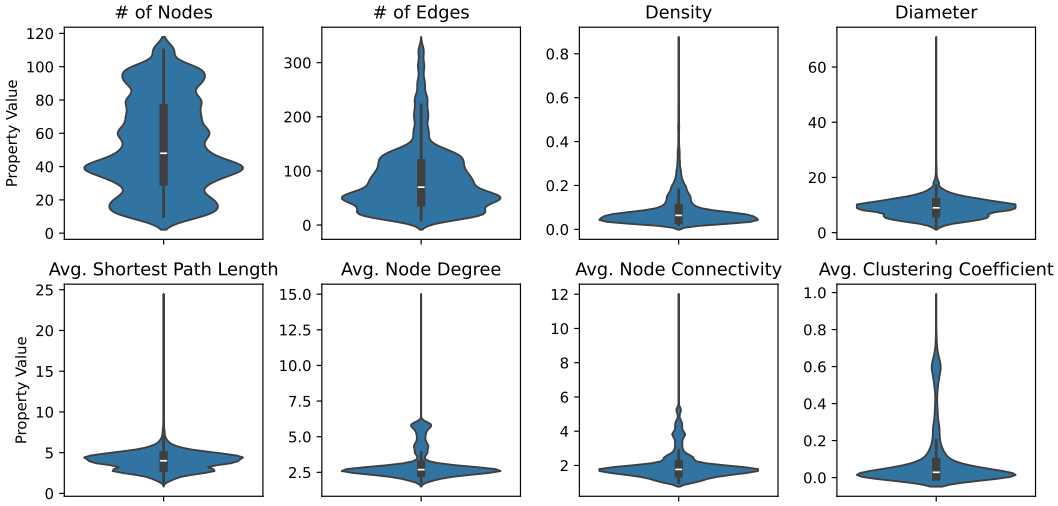


Figure 10 Violin plots showing the distribution of values for 8 common graph properties on the *extended Mooney dataset*.

B Pairwise Metric Correlations

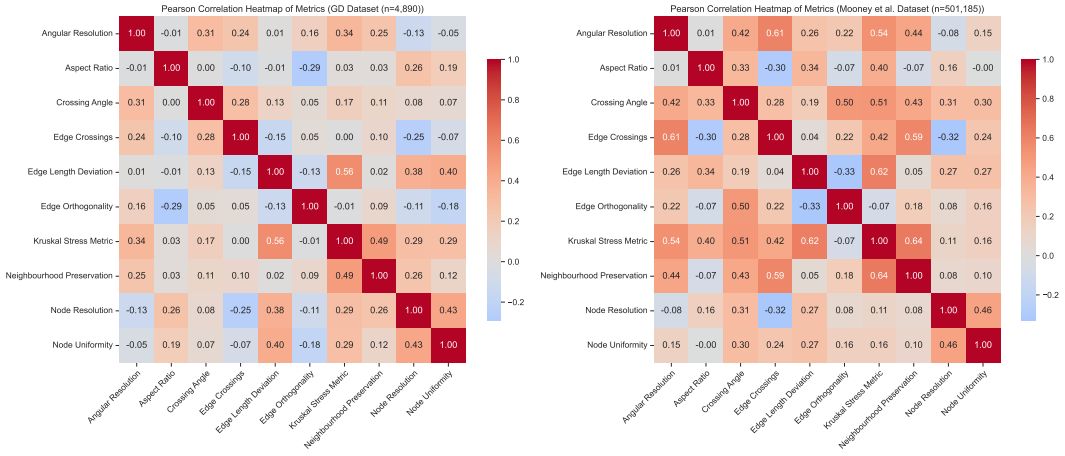


Figure 11 Heatmaps showing the Pearson correlation between pairs of 10 metrics on the *GD-collection-v1* and the extended dataset of 501,185 drawings described by Mooney et al. [18].

C Table of Metric Quartiles

Table 2 Median (M), 1st, and 3rd quartiles for each metric and dataset.

Metric	Stat	Straight	Polygonal	Curved	GD Coll. v1	Mooney
Angular Resolution	Q1	0.348	0.364	0.345	0.352	0.264
	M	0.478	0.516	0.484	0.488	0.482
	Q3	0.639	0.69	0.651	0.657	0.614
Aspect Ratio	Q1	0.474	0.5	0.5	0.499	0.632
	M	0.74	0.733	0.766	0.75	0.834
	Q3	0.889	0.915	0.944	0.92	0.958
Crossing Angle	Q1	0.862	0.773	0.687	0.762	0.631
	M	1.0	1.0	0.995	1.0	0.673
	Q3	1.0	1.0	1.0	1.0	0.807
Edge Crossings	Q1	0.983	0.975	0.937	0.966	0.937
	M	1.0	1.0	0.998	1.0	0.983
	Q3	1.0	1.0	1.0	1.0	0.992
Edge Length Deviation	Q1	0.702	0.611	0.625	0.647	0.637
	M	0.774	0.676	0.685	0.719	0.705
	Q3	0.853	0.749	0.751	0.802	0.812
Edge Orthogonality	Q1	0.48	0.496	0.486	0.486	0.487
	M	0.55	0.592	0.56	0.559	0.517
	Q3	0.68	0.756	0.713	0.708	0.581
Kruskal Stress Metric	Q1	0.636	0.607	0.609	0.618	0.589
	M	0.71	0.675	0.669	0.686	0.669
	Q3	0.798	0.743	0.736	0.766	0.74
Neighbourhood Preservation	Q1	0.3	0.379	0.372	0.341	0.115
	M	0.484	0.5	0.5	0.5	0.239
	Q3	0.643	0.652	0.655	0.65	0.387
Node Resolution	Q1	0.077	0.051	0.067	0.067	0.019
	M	0.136	0.101	0.131	0.127	0.036
	Q3	0.222	0.183	0.209	0.207	0.065
Node Uniformity	Q1	0.545	0.455	0.455	0.5	0.468
	M	0.633	0.586	0.6	0.6	0.551
	Q3	0.727	0.714	0.714	0.714	0.622

From Local Pair-Crossing Number to Local Crossing Number

Jacob Fox
Department of Mathematics, Stanford University, CA, USA

János Pach
HUN-REN Alfréd Rényi Institute of Mathematics, Budapest, Hungary

Andrew Suk
Department of Mathematics, University of California San Diego, La Jolla, CA, USA

Abstract

We prove that if a graph can be drawn in the plane such that each edge crosses at most k other edges, then it can be redrawn so that each edge participates in at most $k^3 + O(k^2)$ crossings. This improves the previous exponential bound that follows from a result of Schaefer and Štefankovič and answers a question of Ackerman and Schaefer.

2012 ACM Subject Classification Mathematics of computing → Combinatorics

Keywords and phrases Crossing numbers, pair crossing numbers

Digital Object Identifier 10.4230/LIPIcs.GD.2025.31

Funding This manuscript was completed while the authors were in residence at SLMath Berkeley, during the Spring semester of 2025, supported by NSF Grant DMS-1928930.
Jacob Fox: supported by NSF awards DMS-2452737 and DMS-2154129.
János Pach: Rényi Institute, Budapest. Supported by NKFIH grant K-176529, Austrian Science Fund Z 342-N31 and ERC Advanced Grant "GeoScape."
Andrew Suk: Supported by NSF grants DMS-1952786 and DMS-2246847.

Acknowledgements The authors want to express their gratitude to Géza Tóth for directing their attention to the problem discussed in this note, and for his many valuable remarks and suggestions.

1 Introduction

In much of the early literature about Turán's brick factory problem and other questions related to graph drawing, the precise problem setting was ambiguous. The basic notions were thought to be so self-evident, that it was often felt that there was no need for definitions. This anomaly was pointed out by Pach and G. Tóth [11, 12], who made a distinction between several different crossing numbers. The two most important among them were the *crossing number*, $\mathrm{cr}(G)$, and the *pairwise crossing number*, $\text{pair-cr}(G)$, of a graph G. The first one is the minimum possible number of *crossing points* between the edges of G, in a drawing of G, where

1. two edges, including edges that have a common endpoint, are allowed to cross any finite number of times,
2. no three edges meet at the same point,
3. the edges are drawn as simple non-self-intersecting Jordan curves.

On the other hand, $\text{pair-cr}(G)$, is the minimum number of *crossing pairs of edges*, that is, if a pair of edges cross multiple times, it still contributes only 1 to this quantity. Obviously, we have $\text{pair-cr}(G) \leq \mathrm{cr}(G)$ for every graph G. It was often assumed with no explanation whatsoever that these two parameters coincide. It is one of the most annoying open problems in the area to decide if this is indeed the case.


© Jacob Fox, János Pach, and Andrew Suk;
licensed under Creative Commons License CC-BY 4.0

▶ **Problem 1** ([11, 12]). *Is it true that* pair-cr$(G) =$ cr(G) *for every graph* G*?*

It was proved in [12] that $\text{cr}(G) \leq \binom{\text{pair-cr}(G)}{2}$ holds for every graph G. This result was subsequently improved several times [6, 7, 18, 20, 21, 22]. The best presently known bound, $\text{cr}(G) = O((\text{pair-cr}(G))^{3/2})$ is due to O. Solé Pi [13].

The separation of the corresponding two parameters can also be observed at a local level. It was probably Ringel [14, 15], who first studied loc-cr(G), the *local crossing number*, of a graph G. It is defined as the minimum number k such that G can be drawn in such a way that the number of crossings along every edge is at most k. As above, we can also define loc-pair-cr(G), the *local pair-crossing number* of G, by counting the maximum number of different *edges* that cross a given edge, without multiplicities. In [2], Ackerman and Schaefer showed that if loc-pair-cr$(G) \leq 2$, then loc-pair-cr$(G) =$ loc-cr(G). Slightly modifying the example depicted on Figure 1 of [9] (see also Exercise (7.1) in [16]), we can construct a graph G with $4 =$ loc-pair-cr$(G) <$ loc-cr$(G) = 5$, showing that these two parameters are not always equal.

Graphs with local crossing number at most k are often called *k-planar*. For getting the best constant in the crossing lemma of Ajtai, Chvátal, Newborn, Szemerédi [3] and Leighton [8], it was useful to estimate the maximum number of edges that a k-planar graph with n vertices can have, for $k \leq 4$; see [1, 10, 9]. The crossing lemma, in turn, has many applications in discrete geometry, in additive combinatorics and elsewhere [19, 4].

A drawing of a graph is called *simple* (or a *simple topological graph*) if any two edges cross at most once. Hoffmann, Liu, Reddy, and C. D. Tóth [5] proved that there exists a function $f(k) \leq (3 + o(1))^k$ such that, if loc-cr$(G) \leq k$, then G also has a simple drawing in which every edge has at most $f(k)$ crossings.

In their seminal paper [17], Schaefer and Štefankovič proved the following (see also Lemma 9.2 in [16]). Suppose that the local pair-crossing number of G satisfies loc-pair-cr$(G) \leq k$. Then one can redraw G in such a way that the total number of crossings along any edge is at most 2^k. This implies that we have loc-cr$(G) \leq 2^k$; see Remark 1 in the last section. In [2], Ackerman and Schaefer asked whether or not this bound can be improved. The aim of this note is to replace this by a polynomial upper bound.

Let $G = (V, E)$ be a graph drawn in the plane. For each vertex $v \in V$, consider the cyclic order of the edges around v. The system of these cyclic permutations is called the *rotation system* of the drawing.

▶ **Theorem 2.** *Let G be a graph that can be drawn in the plane such that every edge crosses at most k other edges, that is,* loc-pair-cr$(G) \leq k$*. Then we have*

$$\text{loc-cr}(G) \leq k(k+1)(k+2).$$

That is, G can be redrawn in such a way that along every edge, the number of crossings is at most $k(k+1)(k+2)$. We can also ensure that the rotation systems of the initial and final drawings are the same.

Combining this result with the theorem of Hoffmann et al. [5] mentioned above, we obtain the following.

▶ **Corollary 3.** *Suppose that the local pair-crossing number of a graph G is at most k. Then G has a simple drawing such that along every edge there are at most $3^{k^3+O(k^2)}$ crossings.*

2 Proof of Theorem 2

We use standard terminology; see, e.g., [16]. Let $G = (V, E)$ be a graph, which may have loops and multiple edges. A *drawing* of G, denoted by $\mathcal{D}(G)$, is a representation of G in which the vertices are distinct points in the plane, and the edges are Jordan arcs connecting the corresponding pairs of points. Whenever there is no danger of confusion, in notation and terminology, we make no distinction between the vertices and the points representing them, and the edges and the corresponding Jordan arcs. No edge is allowed to pass through a vertex, no three edges pass through the same point, and any two edges have only finitely many points in common. We may assume that no two edges "touch," that is, if they have an interior point in common, then they must properly cross at this point. Otherwise we can locally pull apart the two edges near their touching point to get rid of the touching. We may also assume that there is no edge that crosses itself, otherwise we can shorten it by deleting the resulting loops.

Proof of Theorem 2. Let G be a graph on n vertices and m edges, $e_1, \ldots, e_m \in E(G)$, and let $\mathcal{D}(G)$ be a drawing of G in the plane such that each edge in G crosses at most k other edges. Let $\mathcal{R}$ be the rotation system of the drawing $\mathcal{D}(G)$. If G is not connected, we can pull apart the drawing $\mathcal{D}(G)$ so that the connected components do not interact with one another. So, we may assume without loss of generality that G is a connected graph, and let T be a spanning tree of G. Furthermore, let us assume that $T = \{e_1, \ldots, e_{n-1}\}$, and for each $i \in [n-1]$, the edges $e_1, \ldots, e_i$ form an edge-induced connected subgraph of G. We will need the following lemma.

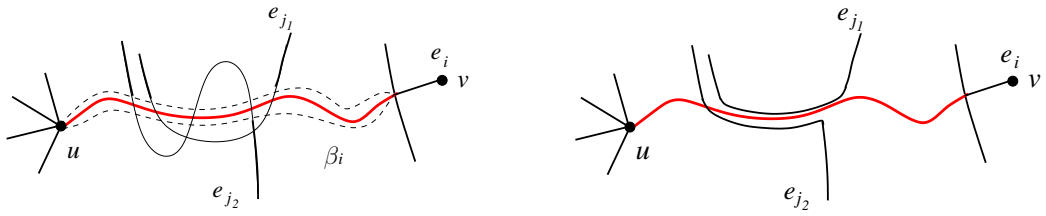
▶ **Lemma 4.** *Let $T \subset G$ and $\mathcal{D}(G)$ be as described above. Then for each $e_i \in T$, there is a subcurve $\alpha_i \subset e_i$ in the plane such that*

1. *no two subcurves α_i, α_j cross each other properly (but one may have an endpoint on the other),*
2. *every vertex in G is an endpoint of some curve α_i, and*
3. *the union $\mathcal{T} = \bigcup_{i=1}^{n-1} \alpha_i$ is a simply connected set in the plane.*

Proof. For each $i \in [n-1]$, we will greedily construct the subcurves $\alpha_1, \ldots, \alpha_i$ with the properties described above, such that for $\mathcal{T}_i = \bigcup_{j=1}^{i} \alpha_j$, $\mathcal{T}_i$ is simply connected and contains all of the endpoints of the edges $e_1, \ldots, e_i$.

We start with $\alpha_1 = e_1$. Having defined $\alpha_1, \ldots, \alpha_{i-1}$ with the desired properties, we obtain α_i as follows. Consider edge $e_i \in T$. Since the edges $e_1, \ldots, e_{i-1}$ form an edge-induced connected subgraph of G, and the drawing $\mathcal{T}_{i-1} = \bigcup_{j=1}^{i-1} \alpha_j$ contains the endpoints of $e_1, \ldots, e_{i-1}$, exactly one of the endpoints of e_i lies in $\mathcal{T}_{i-1}$. Let u be the endpoint of e_i that does lie in $\mathcal{T}_{i-1}$ and v be the other endpoint that does not lie in $\mathcal{T}_{i-1}$. Then we define α_i by starting at vertex v and tracing along the curve e_i until we reach a point on $\mathcal{T}_{i-1}$, which is either a crossing point or a vertex. Clearly, $\mathcal{T}_i = \mathcal{T}_{i-1} \cup \alpha_i$ is simply connected and contains the endpoints of the edges $e_1, \ldots, e_i$. ◀

Let $\alpha_i \subset e_i$, $1 \le i \le n-1$, be the subcurves from Lemma 4, and let $\mathcal{T} = \bigcup_{i=1}^{n-1} \alpha_i$. We then define γ to be a simple closed curve drawn very close to $\mathcal{T}$, such that $\mathcal{T}$ lies inside of γ, that is, $\mathcal{T}$ lies in the closed connected component of $\mathbb{R}^2 \setminus \gamma$. Moreover, γ will not contain any crossing points in our drawing $\mathcal{D}(G)$, and apart from the crossing points on $\mathcal{T}$, all other crossing points will lie outside of γ. For each subcurve $\alpha_i \in \mathcal{T}$, let β_i be a simple closed curve drawn very close to α_i, such that β_i lies inside of γ, the endpoints of α_i lie on β_i, and all crossing points along α_i lie inside of β_i. Moreover, for $j \neq i$, the crossing points along the



(a) Before redrawing. **(b)** After redrawing.

Figure 1 For edge $e_i = uv$, α_i drawn in red, and β_i drawn dashed, we redraw edges e_{j_1} and e_{j_2} inside of β_i.

curve α_j lie outside of β_i. Hence, either β_i and β_j are disjoint, or they are tangent if α_i and α_j have a common endpoint, or β_i and β_j have two points in common if an endpoint of α_i lies in the interior of α_j (or vice versa).

By construction, for each subcurve α_i, there are $t \leq k$ edges $e_{j_1}, \ldots, e_{j_t} \in E(G)$ that cross α_i. Inside of β_i, we can redraw each edge $e_{j_1}, \ldots, e_{j_t}$ so that it intersects the curve α_i at most once as follows. By starting at one of the endpoints of e_{j_r}, we trace along e_{j_r} until it intersects β_i for the first time, which is right before it would intersect α_i. We denote this subcurve as f_{j_r} (the first part of e_{j_r}). Likewise, by starting at the other endpoint of e_{j_r}, we again trace along e_{j_r} until it intersects β_i for the first time, and denote this subcurve as ℓ_{j_r} (the last part of e_{j_r}). We then delete the part of e_{j_r} between f_{j_r} and ℓ_{j_r} (the middle part of e_{j_r}), and then connect f_{j_r} and ℓ_{j_r} by drawing a curve inside of β_i from the endpoint of f_{j_r} on β_i to the endpoint of ℓ_{j_r} on β_i. Clearly, this can be done so that this new drawing of e_{j_r} crosses α_i at most once. We can perform this redrawing procedure to each $e_{j_1}, \ldots, e_{j_t}$ such that afterwards, each pair of edges e_{j_r}, $e_{j_w} \in \{e_{j_1}, \ldots, e_{j_t}\}$ will have at most one crossing point between them inside of β_i, and each such edge e_{j_r} crosses α_i at most once. See Figure 1. After applying this redrawing procedure for each α_i, let $\mathcal{D}'(G)$ denote the new drawing of G in the plane. Hence, no additional crossings were created outside of γ in $\mathcal{D}'(G)$, and the rotation system $\mathcal{R}$ in the new drawing has not changed.

Given $\mathcal{D}'(G)$, we now redraw the edges of G outside of γ as follows. First notice that for each edge $e_i \in E(G)$, e_i will cross γ at most $2k+2$ times in $\mathcal{D}'(G)$. Indeed, starting at one of the endpoints of e_i, we move along e_i and first cross γ without crossing $\mathcal{T}$. Continuing along e_i, since e_i will cross at most k subcurves α_i, we have another possible $2k$ crossing points on γ. Finally, as we approach the other endpoint of e_i, we cross γ one last time for a total of at most $2k+2$ crossing points along γ. Hence, outside of γ, each edge e_i will give rise to at most $k+1$ pairwise disjoint *outer arcs*, with end points on γ. See Figure 2. Hence, all of the edges in G will give rise to at most $(k+1)|E(G)|$ outer arcs outside of γ. Given the circular structure of the endpoints of these outer arcs by γ, we can redraw all of the outer arcs outside of γ such that any two outer arcs cross at most once, and the endpoints of each outer arc remain the same. Moreover, if edges e_i and e_j did not cross outside of γ, then they still do not cross outside of γ in the new drawing. After we redraw the outer arcs as described above, let $\mathcal{D}''(G)$ be the new and final drawing of G in the plane, and notice that the rotation system of $\mathcal{D}''(G)$ is the same as the rotation system of $\mathcal{D}(G)$. Also notice that since each edge consists of at most $k+1$ outer arcs, two edges e_i and e_j will create at most $(k+1)^2$ crossing points outside of γ in $\mathcal{D}''(G)$.

Finally, we show that each edge e_i in the final drawing $\mathcal{D}''(G)$ has at most $k(k+1)(k+2)$ crossing points. By the above, the number of crossing points on e_i outside of γ is at most $k(k+1)^2$, since e_i crosses at most k other edges, and any two will create at most $(k+1)^2$ crossing points outside of γ. In order to bound the number of crossing points on e_i inside of γ, let us consider the following two cases.

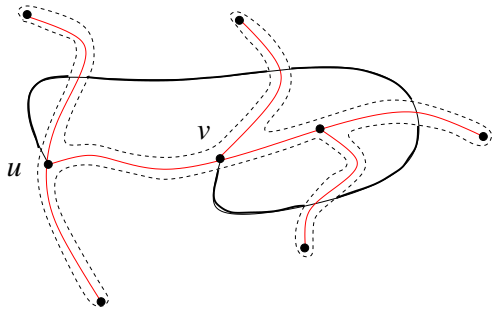


Figure 2 Example of an edge e with endpoints u and v, consisting of 5 outer arcs in bold.

Case 1. Suppose that $e_i \notin T$. Recall that there are at most k subcurves α_j that cross e_i, and for each subcurve α_j, there are at most $k-1$ other edges that cross α_j. Hence, e_i will have at most k crossing points inside of β_j, giving us a total of at most k^2 crossing points on e_i inside of γ.

Case 2. Suppose $e_i \in T$. Again, there are at most k subcurves α_j that cross e_i, and for each subcurve α_j with $j \neq i$, there are at most $k-1$ other edges that cross α_j. Moreover, e_i could have an additional k crossing points along α_i inside of β_i. Hence, e_i will have at most $k^2 + k$ crossing points inside of γ.

Putting everything together, each edge in $\mathcal{D}''(G)$ has at most $k(k+1)^2 + k^2 + k = k(k+1)(k+2)$ crossings. ◀

3 Concluding remarks

1. Suppose G has local pair-crossing number at most k. If there is an edge with more than 2^k crossings on it, then by applying Lemma 9.2 in [17], we can simplify the drawing without creating any new pair of crossing edges (that is, the local pair-crossing number remains at most k), and without increasing the number of crossings along any edge. Repeating this step for each edge of G, we end up with a drawing of G having local crossing number at most 2^k.

2. In the forthcoming journal version of this note, we show that it is possible to improve the bound in Theorem 2 to $\text{loc-cr}(G) \leq \text{loc-pair-cr}(G)^{2.5+o(1)}$ by a more involved argument.

3. Using the redrawing technique used in the proof of Theorem 2, one can show that for any multigraph G, $\text{cr}(G) = O(\text{pair-cr}(G)^{5/3})$. Although this is weaker than the result stated in the introduction due to Solé Pi, we plan to include the proof in the forthcoming journal version for the interested reader.

References

1 E. Ackerman. On topological graphs with at most four crossings per edge. *Comput. Geom.*, 85:101574, 2019. doi:10.1016/J.COMGEO.2019.101574.

2 E. Ackerman and M. Schaefer. A crossing lemma for the pair-crossing number. In *Graph Drawing and Network Visualization: 22nd International Symposium, GD 2014, Würzburg, Revised Selected Papers*, pages 222–233. Springer, 2014. doi:10.1007/978-3-662-45803-7_19.

3 M. Ajtai, V. Chvátal, M. Newborn, and E. Szemerédi. Crossing-free subgraphs. In *Theory and Practice of Combinatorics, North-Holland Math. Stud.*, pages 9–12. North-Holland, Amsterdam, 1982.

4 G. Elekes. On the number of sums and products. *Acta Arithmetica*, 81(4):365–367, 1997.

5 M. Hoffmann, C.-H. Liu, M. M. Reddy, and C. D. Tóth. Simple topological drawings of k-planar graphs. In *Graph Drawing and Network Visualization: 28nd International Symposium, GD 2020, Vancouver, Revised Selected Papers*, pages 390–402. Springer, 2020. doi:10.1007/978-3-030-68766-3_31.

6 J. Karl and G. Tóth. Crossing lemma for the odd-crossing number. *Comput. Geom.*, 108:101901, 2023. doi:10.1016/J.COMGEO.2022.101901.

7 J. R. Lee. Separators in region intersection graphs. *arxiv:1608.01612*, 2016.

8 F.T. Leighton. Complexity issues. In *VLSI: Optimal Layouts for the Shuffle-Exchange Graph and Other Networks*. MIT Press, Cambridge, MA, 1983.

9 J. Pach, R. Radoičić, G. Tardos, and G. Tóth. Improving the crossing lemma by finding more crossings in sparse graphs. *Discrete Comput. Geom.*, 36:527–552, 2006. doi:10.1007/S00454-006-1264-9.

10 J. Pach and G. Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17:427–439, 1997. doi:10.1007/BF01215922.

11 J. Pach and G. Tóth. Thirteen problems on crossing numbers. *Geombinatorics*, 9:194–207, 2000.

12 J. Pach and G. Tóth. Which crossing number is it anyway? *J. Combin. Theory Ser. B*, 80:225–246, 2000. doi:10.1006/JCTB.2000.1978.

13 O. S. Pi. Pair crossing number, cutwidth, and good drawings on arbitrary point sets. *Discrete Comput. Geom.*, 73:310–326, 2025. doi:10.1007/S00454-024-00708-Z.

14 G. Ringel. Ein Sechsfarbenproblem auf der Kugel. *Abh. Math. Sem. Univ. Hamburg*, 29:107–117, 1965.

15 G. Ringel. A nine color theorem for the torus and the Klein bottle. In *The theory and applications of graphs (Kalamazoo, Mich., 1980)*, pages 507–515. Wiley, New York, 1981.

16 M. Schaefer. *Crossing Numbers of Graphs*. CRC Press, 2017.

17 M. Schaefer and D. Štefankovič. Decidability of string graphs. *J. Comput. System Sci.*, 68:319–334, 2004. doi:10.1016/J.JCSS.2003.07.002.

18 J. Matoušek. Near-optimal separators in string graphs. *Combin. Probab. Comput.*, 23:135–139, 2014. doi:10.1017/S0963548313000400.

19 J. Spencer, E. Szemerédi, and W. Trotter. Unit distances in the Euclidean plane. In *Graph theory and combinatorics (Cambridge, 1983)*, pages 293–303. Academic Press, London, 1984.

20 G. Tóth. Note on the pair-crossing number and the odd-crossing number. *Discrete Comput. Geom.*, 39:791–799, 2008. doi:10.1007/S00454-007-9024-Z.

21 G. Tóth. A better bound for the pair-crossing number. In *Thirty Essays on Geometric Graph Theory*, pages 563–567. Springer, New York, NY, 2013.

22 P. Valtr. On the pair-crossing number. In *Combinatorial and Computational Geometry, Math. Sci. Res. Inst. Publ.*, pages 569–575. Cambridge Univ. Press, Cambridge, 2005.

Planar Stories of Graph Drawings: Algorithms and Experiments

Carla Binucci
University of Perugia, Italy

Walter Didimo
University of Perugia, Italy

Eleni Katsanou
National Technical University of Athens, Greece

Antonios Symvonis
National Technical University of Athens, Greece

Sabine Cornelsen
University of Konstanz, Germany

Seok-Hee Hong
University of Sydney, Australia

Maurizio Patrignani
Roma Tre University, Italy

Samuel Wolf
University of Würzburg, Germany

Abstract

We address the problem of computing a dynamic visualization of a geometric graph G as a sequence of frames. Each frame shows only a portion of the graph but their union covers G entirely. The two main requirements of our dynamic visualization are: (i) guaranteeing drawing stability, so to preserve the user's mental map; (ii) keeping the visual complexity of each frame low. To satisfy the first requirement, we never change the position of the vertices. Regarding the second requirement, we avoid edge crossings in each frame. More precisely, in the first frame we visualize a suitable subset of non-crossing edges; in each subsequent frame, exactly one new edge enters the visualization and all the edges that cross with it are deleted. We call such a sequence of frames a *planar story* of G. Our goal is to find a planar story whose minimum number of edges contemporarily displayed is maximized (i.e., a planar story that maximizes the minimum frame size). Besides studying our model from a theoretical point of view, we also design and experimentally compare different algorithms, both exact techniques and heuristics. These algorithms provide an array of alternative trade-offs between efficiency and effectiveness, also depending on the structure of the input graph.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Mathematics of computing → Graph theory; Theory of computation → Graph algorithms analysis

Keywords and phrases Graph Drawing, Dynamic Graphs, Graph Stories, Heuristics, ILP

Digital Object Identifier 10.4230/LIPIcs.GD.2025.32

Related Version *Full Version*: `https://arxiv.org/abs/2508.17532` [5]

Supplementary Material *Dataset*: `https://github.com/ekatsanou/edge-based-graph-stories`

Funding Carla Binucci and Walter Didimo were partially supported by the Italian Ministry of University and Research (MUR) under the PRIN Project no. 2022TS4Y3N – EXPAND and the PON Project RASTA, ARS01_00540; Maurizio Patrignani was partially supported by the Italian Ministry of University and Research (MUR) under PRIN Project no. 2022TS4Y3N - EXPAND.

Acknowledgements This work started at the Bertinoro Workshop on Graph Drawing BWGD 2025.

1 Introduction

Conveying the structure of a graph in a single visualization is the goal of many graph drawing algorithms. Among these, force-directed algorithms are the most popular and widely used in practice [28]. They produce layouts in which vertices are represented as points in the plane and edges as straight-line segments. However, when a graph is complex or locally dense, the layout may feature numerous edge crossings, which significantly reduce its readability, as also witnessed by several human cognitive experiments in graph drawing [34, 35, 36, 37].



© Carla Binucci, Sabine Cornelsen, Walter Didimo, Seok-Hee Hong, Eleni Katsanou, Maurizio Patrignani, Antonios Symvonis, and Samuel Wolf;
licensed under Creative Commons License CC-BY 4.0

This scenario naturally motivates strategies that either attempt to simplify the (single) graph visualization, for instance through graph sampling [16, 24, 32, 38, 41, 42] or edge bundling [2, 23, 31], or that attempt to "distribute" the nodes and edges of the graph across multiple visualizations, i.e., a sequence of frames, each showing only a portion of the graph (see, e.g., [7, 9, 13, 15]). While the first type of strategy focuses on a static visualization that may cause ambiguity or loss of information, computing a sequence of visualization frames allows for a dynamic exploration of the entire graph, without altering its structure. However, it should guarantee *drawing stability* (i.e., the layout of the portion of the graph shared by two consecutive frames should remain unchanged) and each frame should have *low visual complexity*, for example by guaranteeing planarity or few edges crossings.

Contribution. In this paper we address the second type of strategy mentioned above. We introduce a new model for the dynamic visualization of a static graph as a sequence of frames. This model assumes that the input is a geometric graph G (i.e., a graph drawn in the plane with straight-line edges, for example by some force-directed algorithm) and aims to generate a sequence of visualization frames such that: (i) the vertices of G are all present in any frame, always at their initial input coordinates; (ii) the initial frame contains a subset of edges that yields a planar subgraph; (iii) in each subsequent frame, exactly one new edge e enters the visualization and the minimum subset of edges of the current frame that together with e violate planarity disappear; (iv) when an edge leaves the visualization, it no longer appears in the future; (v) the union of all frames contains the whole edge set of G. We call such a sequence of frames a *planar story* of G (see Section 3 for a formal definition).

A natural objective function when computing a planar story is that the amount of edges in any frame is not too small. More precisely, we are interested in planar stories whose minimum number of edges contemporarily displayed (across the entire sequence of frames) is maximized. We call such a story *min-frame optimal*. Besides studying our model from a theoretical point of view, we design and experimentally compare different algorithms (both exact techniques and heuristics). Our main results are as follows:

- We show in Section 5 that a min-frame optimal planar story of a geometric graph G can be efficiently computed when G is 2-plane (i.e., each edge is crossed at most twice), although the problem is NP-hard in the general case (Section 4).
- We describe several heuristics based on a common greedy framework; all of them work for general graphs (Section 5). They differ in the strategy for computing the initial and the last frames, and in the criteria used to select the edge that enters the visualization in each subsequent frame. We also describe an Integer Linear Program (ILP) that solves the problem optimally (Section 6).
- We discuss the results of an extensive experimental analysis that compares our heuristics and the exact algorithm. The results highlight algorithmic trade-offs between efficiency and effectiveness, and show how some of our heuristics are able to achieve the optimum in many cases in a short runtime. See Section 7.

The paper concludes by discussing some future research directions (Section 8). Due to space limitations some proofs and technical details can be found in [5].

2 Related Work

The research in this paper is inspired by a recent topic in graph drawing named *graph stories*. A graph story corresponds to a temporal sequence of frames, each displaying only part of the graph and whose union covers all nodes and edges of the graph. The term "graph story"

was introduced by Borrazzo, Da Lozzo, Di Battista, Frati, and Patrignani [9]. Different from our scenario, in their model no drawing of the graph is provided in advance. The vertices enter the visualization one at a time and persist in the visualization for a fixed amount of time (i.e., for a given subsequence of frames). When a vertex enters the visualization its coordinates have to be computed. In any frame, the user sees only the graph induced by the displayed vertices; the drawing in each frame must be straight-line and planar, and the layout of the graph portion shared by two consecutive frames cannot change. The authors in [9] provide bounds on the area requirement of graph stories for paths and trees, and exhibit planar graphs that do not admit a graph story within their rules. The model in [9] has been further studied in [13, 12], by allowing edges to be Jordan curves rather than straight-line segments. Also, several authors proposed a different setting in which each vertex persists in the visualization until all its neighbors have been displayed [6, 7, 17]. In all the aforementioned papers, the main focus is on exploring the time complexity and theoretical questions, often restricting to very specific graph families. Conversely, our model works for general graphs and, besides providing a solid theoretical basis, our goal is to derive practical implementations with different trade-offs between effectiveness and efficiency.

Another problem related to our research has been studied in [14, 15]. Similarly to our setting, it assumes that the input is a straight-line drawing of a graph; the objective is to partition the edge set into the minimum number of frames, such that the drawing in each frame has some desired property (e.g., planarity). However, unlike our model, the visualization frames do not form a temporal sequence and each edge appears in just one frame.

Finally, we remark that representing a graph across a sequence of frames (using small multiples or animation approaches) is strongly related to the problem of visually conveying dynamic graphs (see [4] for a survey on this topic). However, a dynamic graph evolves over time, and the ordered sequence of elements that appear or disappear during the graph evolution is part of the input for the drawing algorithm. In our setting, the graph is static and the algorithm can decide the temporal sequence of edges for dynamically visualizing it.

3 Basic Definitions and Preliminary Considerations

Given a graph G, we denote by $V(G)$ and $E(G)$ the set of vertices and the set of edges of G, respectively. For a set $V' \subseteq V(G)$, let $N(V')$ be the set of neighbors of vertices in V' and let $G - V'$ be the subgraph of G induced by $V(G) \setminus V'$. Let G be a *geometric graph*, that is, each vertex $v \in V(G)$ is a point in the plane and each edge $e = uv \in E(G)$ is the straight-line segment connecting u and v. A *planar frame* F of G is any subgraph of G consisting of all the vertices of G and a subset of edges of G that do not cross. A *planar story* of G is a sequence $\sigma = \langle F_1, F_2, \ldots, F_\tau \rangle$ such that:

1. Each F_i is a planar frame of G, with $i \in \{1, \ldots, \tau\}$.
2. Each edge $e \in E(G)$ appears in a non-empty subsequence of consecutive planar frames.
3. For each $i \in \{2, \ldots, \tau\}$, F_i contains exactly one edge e of G that is not in the union of all frames F_j, with $j < i$, and exactly those edges of F_{i-1} that do not cross e.

Let $\Sigma(G)$ be the set of all planar stories of G. For a planar story $\sigma = \langle F_1, F_2, \ldots, F_\tau \rangle \in \Sigma(G)$, we call F_1 and F_τ the *initial frame* and the *last frame* of σ, respectively, let m_i be the *size* of F_i, that is, the number of edges in F_i. Let $\mu(\sigma) = \min\{m_1, \ldots, m_\tau\}$, and let $\mu(G)$ be the maximum $\mu(\sigma)$ over all planar stories σ of G, i.e., $\mu(G) = \max\{\mu(\sigma) \mid \sigma \in \Sigma(G)\}$. If $\mu(\sigma) = \mu(G)$ then σ is said to be *min-frame optimal*. We call MaxMinFramePlanarStory(G) the problem of computing a min-frame optimal planar story of G.

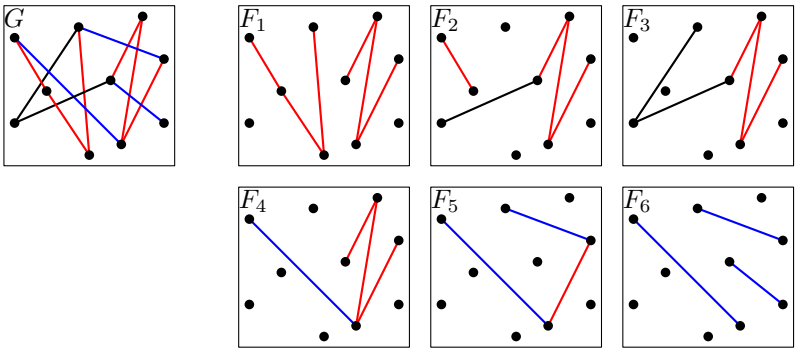


Figure 1 A planar story consisting of 6 frames. The top-leftmost figure shows the input geometric graph G; red edges and blue edges are those in the first frame and in the last frame, respectively. The black edges are the remaining edges.

▶ Remark 1. Since all the crossing-free edges of G appear in every frame of any planar story of G, we can safely disregard them when solving MaxMinFramePlanarStory(G). Hence, without loss of generality, we will assume that G does not contain crossing-free edges.

Fig. 1 shows a geometric graph G and a planar story σ of G, consisting of six frames and where $\mu(\sigma) = 3$. In the figure, the red edges are those of the initial frame, the blue edges are those of the final frame, and the black edges are the remaining edges.

▶ **Proposition 2.** *In any planar story $\sigma = \langle F_1, \ldots, F_\tau \rangle$, F_1 and F_τ are edge disjoint.*

Proof. Let e be an edge in F_1. By Remark 1, e is crossed by some edge e' of the graph. Let F_i be the first frame of σ that contains e', with $i \geq 2$. By definition of planar story, e does not occur in any frame F_j with $j \geq i$. In particular, e is not contained in F_τ. ◀

Given a planar frame F of G, let $\mu_F(G)$ be the maximum $\mu(\sigma)$ over all planar stories σ of G with initial frame F. Proposition 2 implies the following.

▶ **Corollary 3.** *Let F be a planar frame of G, and let F' be a maximum planar subgraph of $G - E(F)$. Then $\mu_F(G) \leq \min\{|E(F)|, |E(F')|\}$. Therefore, $\mu(G) \leq |E(G)|/2$.*

Planar Graph Stories and Crossing Graph. The *crossing graph* X of a geometric graph G is defined as follows: (i) the vertex set $V(X)$ corresponds to the edge set $E(G)$; (ii) there is an edge $uv \in E(X)$ if and only if the edges corresponding to u and v in $E(G)$ cross each other. The crossing-free edges of G correspond to isolated vertices of X; by Remark 1, we assume that X has no isolated vertices. Note that a graph is the crossing graph of a geometric graph if and only if it is a *segment intersection graph* [29]. This includes cliques, but also planar graphs [10, 19]. The crossing graph of a geometric graph with m edges and χ crossings can be computed in $\mathcal{O}(m \log m + \chi)$ time with the algorithm of Balaban [3] for segment intersection.

If $\sigma = \langle F_1, \ldots, F_\tau \rangle$ is a planar story of G, each frame F_i ($i \in \{1, \ldots, \tau\}$) corresponds to an *independent set* $X[F_i]$ of X, i.e., no two vertices of $X[F_i]$ are adjacent. We denote by $\sigma_X = \langle X[F_1], \ldots, X[F_\tau] \rangle$ the sequence that describes the planar story σ from the point of view of X. See Fig. 2 for an example. We will also refer to σ_X as a planar story and to the independent sets $X[F_i]$ as (planar) frames. If not stated otherwise, we use red for the initial frame and blue for the final frame in our figures. A *maximum pair of independent sets* is a pair (I_1, I_2) of two disjoint independent sets such that $\min\{|I_1|, |I_2|\}$ is maximum. A maximum pair (I_1, I_2) of independent sets is *Pareto optimal* if also $\max\{|I_1|, |I_2|\}$ is maximum. As a further corollary of Proposition 2, we obtain the following.

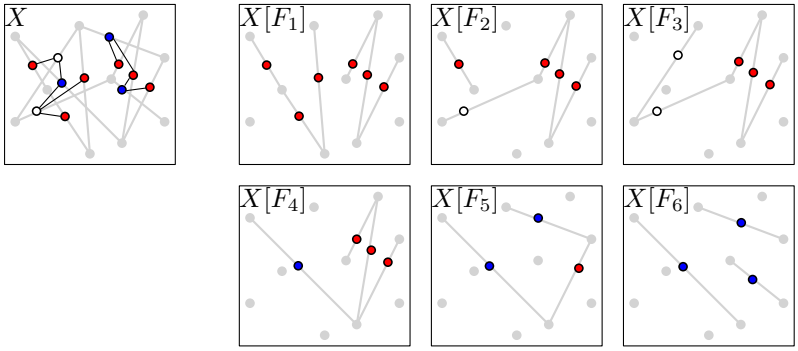
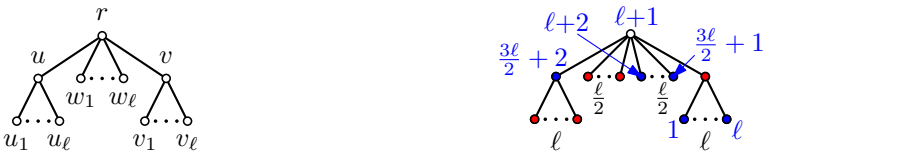
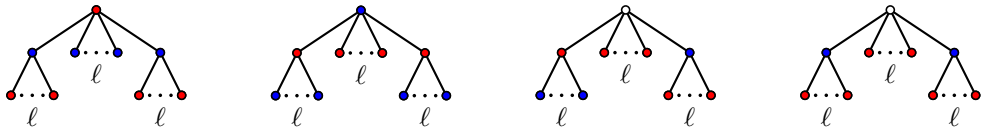


Figure 2 The crossing graph X of the graph G in Fig. 1 (top-left) and the planar story in Fig. 1 from the point of view of the crossing graph X of G.



(a) crossing graph **(b)** non-maximal initial frame, $\mu_F(G) = \mu(G) = \frac{3}{2}\ell + 1$



(c) $\mu_F(G) \leq |I'| = \ell + 2$ **(d)** $\mu_F(G) \leq |I| = \ell + 2$ **(e)** $\mu_F(G) \leq |I'| = \ell + 1$ **(f)** $\mu_F(G) \leq |I'| = 2$

Figure 3 The crossing graph X of some graph G. The set $I \subseteq V(X)$ of red vertices corresponds to the initial frame F. Blue vertices are a maximum independent set I' within $X - I$. (c)-(f) If we choose I as a maximal independent set, then there is a frame with at most $\min\{|I|, |I'|\} \leq \ell + 2$ edges. However, there is a planar story (b) with at least $\frac{3}{2}\ell + 1$ edges in any frame. Numbers close to the vertices indicate the order in which we add the respective edges into the planar story.

▶ **Corollary 4.** *Let G be a geometric graph, X be the crossing graph of G, and (I_1, I_2) be a* maximum pair of independent sets *of X. Then $\mu(G) \leq \min\{|I_1|, |I_2|\}$.*

We observe that, in order to get a min-frame optimal story of a geometric graph G, it might be necessary to choose as the initial frame a planar subgraph of G that is not a maximal planar subgraph. In terms of the crossing graph, this means that it might be necessary to start with an independent set that is not maximal. See Fig. 3 for an example where the crossing graph is a caterpillar.

4 Problem Complexity

In Section 5 we show how to solve MaxMinFramePlanarStory(G) optimally in linear time when G is a 2-plane graph and in cubic time if G is a 3-plane graph whose crossing graph has no cycles. We recall that a geometric graph is *k-plane* if each edge is crossed at most k times. However, MaxMinFramePlanarStory(G) is NP-hard in general. To this end, consider its decision version MaxMinFramePlanarStoryD(G,m): Given a graph G and a target value m, does G admit a planar story $\sigma = \{F_1, F_2, \ldots, F_\tau\}$ such that $\mu(\sigma) \geq m$?

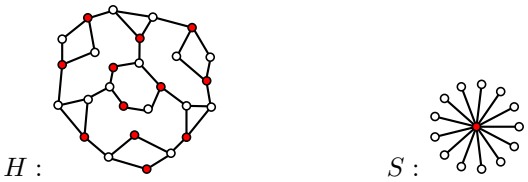


Figure 4 If the size of a maximum independent set of H is k then the size of the maximum frame containing the center of S – indicated by red vertices – is at most $k+1$.

▶ **Theorem 5.** *Problem* MaxMinFramePlanarStoryD(G,m) *is NP-complete.*

Proof. It is easy to see that MaxMinFramePlanarStoryD(G,m) is in NP. To prove the hardness, we apply a reduction from PlanarMaximumIndependentSetD(H,k), i.e., the problem of deciding whether a planar graph H admits an independent set of size at least k; this problem is known to be NP-hard [30, Theorem 4.1]. To this end, let X be the disjoint union of H and a star S with $k+1$ leaves (see Fig. 4). Since X is planar, it is a segment intersection graph [10], i.e., a crossing graph of some geometric graph G. We show that $\mu(G) \geq k+1 =: m$ if and only if H contains an independent set of size at least k.

Assume first that H contains an independent set I of size at least k. Consider the following planar story. The initial frame contains the edges corresponding to I and the center of the star S. The size of the initial frame is $k+1$. For the subsequent frames, we add the edges corresponding to the leaves of S, in any arbitrary order. This increases the size of the frame up to $2k+1$. Finally, we add in arbitrary order all edges corresponding to vertices of H that were not in I. During this process, the edges corresponding to the leaves of S will never be removed. Thus, the size of each frame is at least $k+1$.

Assume now that there is a planar story σ of G with $\mu(\sigma) \geq k+1$. We show that H contains an independent set of size at least k. Let F be the frame of σ that contains the edge corresponding to the center of S. In this frame, there is no edge corresponding to a leaf of S. This implies that frame F must contain at least k non-intersecting edges that correspond to at least k independent vertices of H. ◀

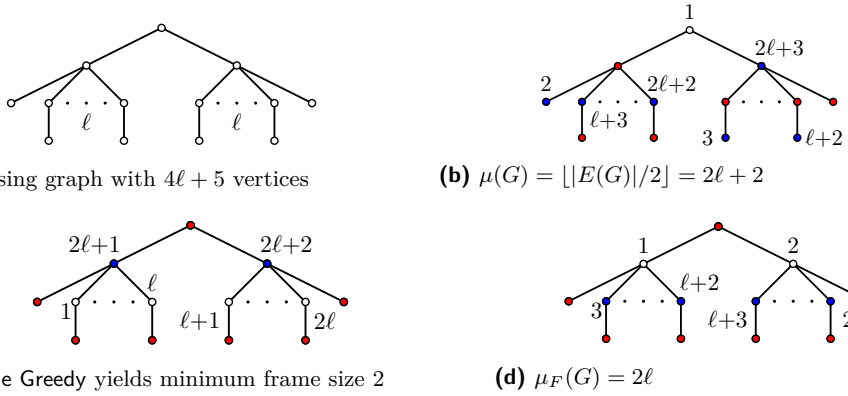
5 A Greedy Heuristic

We devise variants of a greedy heuristic for the problem MaxMinFramePlanarStory(G). The heuristic computes the frames one by one. Throughout the algorithm, we call the edges of the current frame *current edges*, the edges that were already removed *past edges*, and those that still have to be inserted *future edges*. The *current/future degree* of an edge e is the number of current/future edges that cross e. A simple greedy strategy may work as follows.

Simple Greedy

- Phase 1. Start with some initial planar frame F_1, and let $\tau = |E(G)| - |E(F_1)| + 1$ be the total number of frames in the story.
- Phase 2. For each step $i = 1, \ldots, \tau - 1$, let F_i be the current planar frame. Pick a future edge e with the minimum current degree and let F_{i+1} be the frame obtained from F_i by adding e and by removing the edges that are crossed by e.

Since Simple Greedy always first increases an initial frame to a maximal planar subgraph, the example in Fig. 3 implies that this strategy is not optimum in general. Moreover, Simple Greedy does not yield a constant-factor approximation even if the crossing graph is a tree; see the example in Fig. 5. The alternative order in Fig. 5d is better than the solution of Simple



(a) Crossing graph with $4\ell + 5$ vertices

(b) $\mu(G) = \lfloor |E(G)|/2 \rfloor = 2\ell + 2$

(c) Simple Greedy yields minimum frame size 2

(d) $\mu_F(G) = 2\ell$

Figure 5 The crossing graph X of some graph G. The vertex labels indicate the order in which we add the respective edges of G to the initial frame (red vertices). The two cases (c) and (d) in the bottom row start with the same initial frame F.

Greedy, as it ensures that the last frame remains sufficiently large. Hence, we introduce a refined version of the simple greedy strategy, described hereunder. We first describe the general strategy of the two phases of the Advanced Greedy algorithm. In the following we provide details about alternative variants for executing both of them.

Advanced Greedy

- Phase 1. Compute two edge-disjoint planar frames, the initial frame F_1 and the final frame F_τ, where $\tau = |E(G)| - |E(F_1)| + 1$.
- Phase 2. For each step $i = 1, \ldots, \tau - 1$, let F_i be the current planar frame. To make sure that all edges of F_τ are indeed in the final frame, we say that a future edge is *admissible* if it is (i) not in F_τ or (ii) not crossed by any other future edge. We pick an admissible edge e with the minimum current degree. Let F_{i+1} be the frame obtained from F_i by adding e and by removing the edges that are crossed by e.

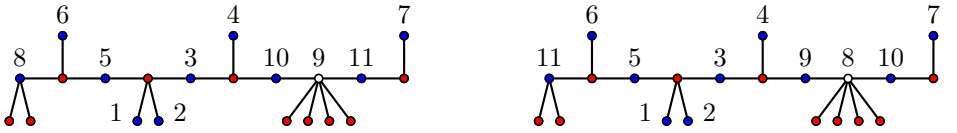
Note that Phase 2 of Advanced Greedy is related to the reconfiguration problem [25] that asks for a transformation between two feasible solutions of a problem such that all intermediate results are also feasible. In particular, [26] considered the problem of traversing between two given independent sets of the same size. However, in [26] at each step exactly one vertex is removed and one vertex is added, and vertices may disappear and reappear several times.

Alternative variants for Phase 1. We propose three alternative strategies for computing the initial frame F_1 and the last frame F_τ, or in other words two possibly large disjoint independent sets I_1 (initial frame) and I_τ (final frame) of the crossing graph X of G.

1a. (I_1, I_τ) is a Pareto optimal maximum pair of independent sets such that $|I_1| \leq |I_\tau|$. See Thm. 6 for the computation.

1b. I_1 is a large independent set of X of size at most $|E(G)|/2$ and I_τ is a maximal independent set of $X - I_1$. More precisely, we use the following approach: While $|I_1| \leq |E(G)|/2 - 1$ and $X - I_1 - N(I_1)$ is not empty, iteratively add a vertex of minimum degree in $X - I_1 - N(I_1)$ to I_1. Then, iteratively add a vertex of minimum degree in $X - I_1 - I_\tau - N(I_\tau)$ to I_τ.

1c. Alternating between $i = 1$ and $i = \tau$, iteratively add a vertex of minimum degree in $X - I_1 - I_\tau - N(I_i)$ (if there exists one) to I_i. If $|I_1| > |I_\tau|$, exchange I_1 and I_τ.



(a) greedy: 10,10,11,11,12,12,13,13,12,9,9,10 **(b)** optimal: 10,10,11,11,12,12,13,13,10,10,11,10

Figure 6 A Pareto-optimal maximum pair of independent sets in the crossing graph as initial and last frame with a non-optimal greedy story and a min-frame optimal story. The edges are added according to the numbers of the vertices in the crossing graph to the initial frame (red vertices). The subcaptions show the sequences of the frame sizes.

Variants 1b and 1c can be implemented in linear time in the size of X by sorting the vertices in buckets according to their degrees. Regarding Variant 1a, while it is NP-hard to find a maximum pair of independent sets in a graph [5], there exists a fixed-parameter tractable (FPT) algorithm parameterized by the treewidth of the graph (Thm. 6). Therefore, Variant 1a can be much slower than Variants 1b and 1c, although it may lead to better solutions. We also remark that Thm. 6 establishes a result of independent interest, which goes beyond the purposes of our specific problem.

A *tree decomposition* [39] of a graph X is a tree $\mathcal{T}$ such that each vertex $\nu \in V(\mathcal{T})$ is associated with a set $V(\nu) \subseteq V(X)$, called the *bag* of ν, such that

(a) $\bigcup_{\nu \in V(\mathcal{T})} V(\nu) = V(X)$,
(b) for each edge $uv \in E(X)$, there is a vertex $\nu \in V(\mathcal{T})$ such that $u, v \in V(\nu)$, and,
(c) for each $v \in V(X)$, the subgraph of $\mathcal{T}$ induced by the vertices $\{\nu \in V(\mathcal{T}) \mid v \in V(\nu)\}$ is connected.

The *width* of a tree decomposition is $\max_{\nu \in V(\mathcal{T})} |V(\nu)| - 1$. The *treewidth* $\mathrm{tw}(X)$ of a graph X is the minimum among the widths of any of its tree decompositions.

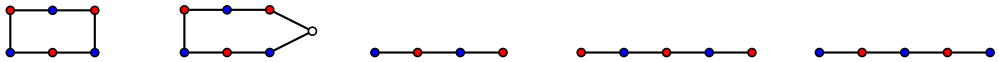
▶ **Theorem 6.** *A Pareto optimal maximum pair of independent sets can be computed in $\mathcal{O}(3^\omega \omega^2 n^3)$ time for a graph X with n vertices and a tree decomposition $\mathcal{T}$ of width ω.*

Proof Sketch: The proof follows the ideas of the FPT approach for constructing a minimum vertex cover as proposed by Niedermeier [33], enhancing it with the concept of Pareto optimal pairs. Root $\mathcal{T}$ at an arbitrary vertex. We use dynamic programming. For a vertex ν of $\mathcal{T}$ let $X(\nu)$ be the subgraph of X induced by the vertices in the bags of the subtree of $\mathcal{T}$ rooted at ν. For each map $c\colon V(\nu) \to \{\text{red, blue, white}\}$, we compute the list $L(\nu, c)$ of Pareto-optimal pairs (α, β) for which $X(\nu)$ contains two disjoint independent sets I_1 and I_2 of size α and β, respectively, such that the red vertices of $V(\nu)$ are contained in I_1, the blue vertices of $V(\nu)$ in I_2, and the white vertices of $V(\nu)$ are neither in I_1 nor I_2. ◀

Given a Pareto-optimal maximum pair (I_1, I_τ) of independent sets, Advanced Greedy 1a does not necessarily result in a planar story with minimum frame size among all planar stories with initial frame I_1 and final frame I_τ. An example is given in Fig. 6: here, there is a vertex that is neither in the initial nor in the final frame whose degree is greater than 3.

▶ **Lemma 7.** *Starting from a Pareto-optimal maximum pair (I_1, I_τ) of independent sets, Advanced Greedy yields a min-frame optimal planar story for a geometric graph G if the crossing graph X contains no cycles and the vertices not in $I_1 \cup I_\tau$ have degree at most three.*

Proof Sketch. The frame sizes are first monotonically increasing and then monotonically decreasing and, thus, at least $\min\{|I_1|, |I_\tau|\}$, which, by Corollary 4, is optimum. ◀



(a) even cycles **(b)** odd cycles **(c)** even paths **(d)** half of odd paths **(e)** other odd paths

Figure 7 The connected components of the crossing graph for a 2-plane graph.

▶ **Theorem 8.** *Advanced Greedy 1a solves* MaxMinFramePlanarStory(G) *optimally*
(a) *in linear time for 2-plane geometric graphs, and*
(b) *in cubic time for those 3-plane geometric graphs for which the crossing graph contains no cycles.*

Proof Sketch. The statement for 3-plane graphs is a corollary of Lemma 7. For 2-plane graphs, the crossing graph has maximum degree two and, thus, its connected components are paths and cycles, and a maximum pair (I_1, I_τ) of independent sets can be computed in linear time. See Fig. 7. Moreover, if we apply Advanced Greedy 1a then the minimum frame size is $\min\{|I_1|, |I_\tau|\}$, unless there is an even cycle but no odd path. In this case, the minimum frame size is $\min\{|I_1|, |I_\tau|\} - 1$. ◀

Alternative variants for Phase 2. In each step of Phase 2, the set E' of admissible future edges with minimum current degree might contain multiple edges. For an edge $e \in E'$, denote by $C(e)$ the set of current edges that cross e. Also, let $E'' \subseteq E'$ be the subset of edges in E' such that, for any $e \in E''$, the number of future edges crossing some edges in $C(e)$ is the maximum over all other edges in E'. We consider two alternative strategies for selecting the next future edge e that enters the drawing: 2a. e is chosen uniformly at random in E'; or 2b. e is chosen uniformly at random in E''.

Variant 2b applies a tie-breaking rule to further restrict the set of admissible edges that can be selected. The rationale is to maximize the amount of future edges that will benefit from the removal of the current neighbors of the edge that will enter the drawing.

6 An ILP Exact Approach

We present an exact integer linear program (ILP) for MaxMinFramePlanarStory(G). Let τ be the number of frames necessary, i.e., $\tau \leq |E(G)|$. For each edge $e \in E(G)$ and each $t \in \{1, \ldots, \tau\}$ we use a binary variable x_e^t to represent that edge e of G is present in frame t. Additionally, we use *lifting variables* $z_e^t \in \{0, 1\}$, where $z_e^t = 1$ indicates that an edge e of G appears in frame t. Finally, we use a variable $y_{\min}$ that corresponds to the minimum number of edges present over all frames.

$$\text{maximize} \quad y_{\min} \tag{1}$$

$$\text{subject to} \quad x_e^t + x_f^t \leq 1 \qquad \forall t \in \{1, \ldots, \tau\}\ \forall \{e, f\} \in E(X) \tag{2}$$

$$\sum_{t=1}^{\tau} x_e^t \geq 1 \qquad \forall e \in E(G) \tag{3}$$

$$\sum_{e \in E(G)} x_e^t \geq y_{\min} \qquad \forall t \in \{1, \ldots, \tau\} \tag{4}$$

$$\sum_{t=1}^{\tau} z_e^t = 1 \qquad \forall e \in E(G) \tag{5}$$

$$z_e^t + x_e^{t-1} \geq x_e^t \qquad \forall e \in E(G)\ \forall t \in \{2, \ldots, \tau\} \tag{6}$$

$$z_e^1 \geq x_e^1 \qquad \forall e \in E(G) \tag{7}$$

$$\sum_{e \in E(G)} z_e^t \leq 1 \qquad \forall t \in \{2, \ldots, \tau\} \tag{8}$$

$$x_e^t, z_e^t \in \{0, 1\} \tag{9}$$

$$y_{\min} \geq 0 \tag{10}$$

Constraints 2 and 3 ensure that no two edges cross in the same frame and that each edge appears in some frames, respectively. Constraints 4 are a linearization of the objective function.

Additionally, we need to model the following property. If an edge e is present in frame F_i and present in frame F_j with $i < j$, then e also needs to be present in all intermediate frames. This can be directly translated, but requires a cubic number of constraints for each edge. Hence, we introduce Constraints 5–7 that model the same property. Consider an edge e that appears at frame t for the first time. Due to Constraints 6 the lifting variable z_e^t must be set to 1 if the edge e appears in frame t, as $x_e^{t-1} = 0$ or since $t = 1$ (Constraints 7). The consecutive frame $t + 1$ can now include this edge, as $x_e^t = 1$ and the variable z_e^{t+1} is no longer required. However, once e disappears in a frame $j > t$, i.e., $x_e^j = 0$, this is no longer possible, as the lifting variables cannot be used anymore due to Constraints 5.

Finally, we include Constraints 8 to enforce that at most one edge appears in each frame, except for the initial frame in which multiple edges may appear.

7 Experimental Analysis

We present the results of an extensive experimental analysis, whose goal is to compare the effectiveness and the efficiency of our heuristics and of the exact algorithm. For the heuristics, we denote by AG-1x2y the variant of algorithm Advanced Greedy that uses Variant 1x for Phase 1 and Variant 2y for Phase 2, where x can be a, b, or c, while y can be a or b. In total we have 6 possible variants of our heuristic. In the following we describe the graph benchmark used for the experiments, the experimental setting, and the obtained results.

Graph Benchmark. To evaluate the performance of our algorithms on different types of graphs, we used several test suites, totaling 637 instances[1]. We considered two main types of instances: Geometric Graphs and Crossing Graphs.

Geometric Graphs. A collection of both random and real graphs drawn with a force-directed algorithm. The different graph classes are as follows

- `random graphs`: 200 geometric graphs: For each $n \in \{10, 20, \ldots, 100\}$ and for each density[2] $d \in \{1.2, 1.6, 2.0, 2.4\}$, we randomly generated 5 distinct graphs with n vertices and $m = d \cdot n$ edges, with Erdős–Rényi's model, which guarantees uniform probability distribution. For each generated graph we computed a straight-line drawing through the popular Fruchterman–Reingold force-directed algorithm [18]. We exploited the NetworkX Python library [22] both for generating the graphs and for computing their drawings. The resulting sizes of the crossing graphs is shown in Fig. 8.

[1] To allow replicability, we made the whole graph benchmark publicly available at `https://github.com/ekatsanou/edge-based-graph-stories`.

[2] The *density* of a graph is the number of its edges divided by the number of its vertices.

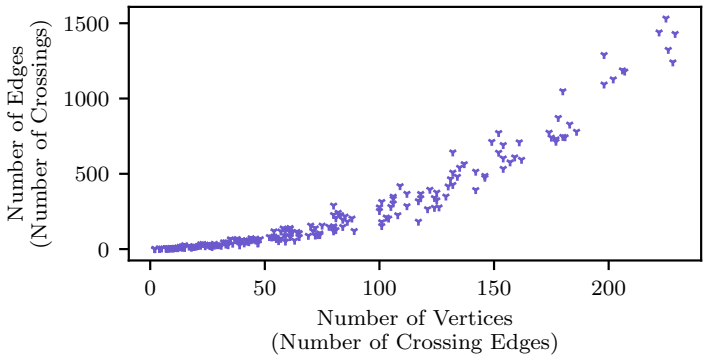


Figure 8 The sizes of the crossing graphs for the `random graphs`. The x-axis is the number of vertices of the crossing graph (i.e., the number of crossing edges of the instance); the y-axis is the number of edges of the crossing graph (i.e., the number of crossings in the instance).

- `real graphs`: 12 graphs with 30 to 379 vertices representing data from various real-world domains (see Table 1).They usually have higher density than the `random graphs` and, consequently, more edge crossings. All these graphs have been taken from the well-known Network Repository [40] (`https://networkrepository.com`). For each graph, we removed self-loops (if any) and computed a straight-line drawing with the implementation of Fruchterman–Reingold's force-directed algorithm in NetworkX.

Table 1 Real graphs with n vertices, m edges, and χ edge-crossings.

Graph name	Category	n	m	χ
insecta-beetle-group-c1-period-1	animal social science network	30	185	1,737
road-chesapeake	road network	39	170	1,049
eco-stmarks	eco network	54	353	6,320
lesmis	miscellaneous network	77	254	838
ca-sandi_auths	collaboration network	86	124	8
gd06-theory	miscellaneous network	101	190	1,015
polbooks	miscellaneous network	105	441	2,465
adjnoun	miscellaneous network	112	425	6,868
rajat11	miscellaneous network	135	680	290
email-enron-only	email network	143	623	5,230
bwm200	miscellaneous	200	596	7
ca-netscience	collaboration network	379	914	901

Crossing Graphs. We generate different types of crossing graphs that are planar. Recall that planar graphs are segment intersection graphs [10, 19]. Therefore, they always reflect the structure of some geometric graphs in terms of edge crossings.

- `caterpillars`: 60 n-vertex caterpillar trees, 10 instances chosen uniformly at random for each value of $n \in \{10, 20, \ldots, 60\}$. First, we selected the diameter δ based on the probability of an n-vertex caterpillar to have such diameter; then we chose how to randomly attach the $n - \delta - 1$ leaves to the $\delta - 1$ internal vertices of the caterpillar [27].

We rejected with probability 0.5 non-symmetric caterpillars, as they would have been generated twice. Due to the large integers involved in the computation, this method allowed us to generate caterpillars with up to 60 vertices.

- `trees`: 80 general trees, with $n \in \{10, 20, \ldots, 80\}$ vertices; for each fixed n we collected 10 distinct instances, generated uniformly at random with the algorithm in [1]
- `series-parallel graphs`: 105 series-parallel graphs, with $n \in \{10, 20, \ldots, 70\}$ vertices and density $d \in \{1.2, 1.4, 1.6\}$. For each pair (n, d) we collected 5 instances, by repeatedly generating biconnected planar graphs using the OGDF library [11] (`https://ogdf.uos.de/`), and discarding those graphs that were not series-parallel.
- `planar graphs`: 180 general connected planar graphs, with $n \in \{10, 20, \ldots, 90\}$ and density $d \in \{1.2, 1.6, 2.0, 2.4\}$. Precisely, we generated 5 instances for each pair (n, d), still using the random generator available in the OGDF library [11].

Since `trees` and `caterpillars` have treewidth 1 and the treewidth of `series-parallel graphs` is 2, these instances are of particular interest for evaluating the performances of the heuristics AG-1a2y, which works in polynomial-time for crossing graphs of bounded treewidth.

Experimental setting. We performed the experiments on a Windows 11 Pro machine with an Intel Core i5-12500 CPU and 16 GB RAM. We used Gurobi [21] version 12.0.1 as ILP solver that used all available cores of the CPU. We implemented the heuristics in Python, using the NetworkX library [22].

On those instances for which the exact algorithm was able to finish the computation within the given time (see below), we compared the optimum with the value computed by the various heuristics. Precisely, we measured the ratio between the minimum frame size of the story computed by each heuristic and the minimum frame size of the story computed by the exact algorithm. With a slight abuse of terminology, we call this value the *approximation ratio*. For the `random graphs` and `real graphs`, the computation of the approximation ratio is restricted to the subset of crossing edges, because crossing-free edges persist in all frames of a planar graph story, regardless of the algorithm used to compute it.

For each instance, we also measured the runtime of each algorithm. We fixed an upper bound of $T = 15$ minutes for the runtime allowed for the exact algorithm, and we recorded the value of the best solution achieved within T (i.e., the incumbent solution). For those instances for which the runtime of the exact algorithm exceeded T, we also measured the optimality gap, that is, the *gap* between the incumbent solution (IS) and the upper bound (UB) estimated by the ILP solver at time T. In formula: $gap = \frac{UB-IS}{UB}$.

Furthermore, after verifying that, for all instances optimally solved by the exact algorithm, the heuristics were also able to produce a solution in less time, we set a maximum time limit of $T' = 10$ minutes for the heuristics based on Variant 1a, whose runtime may grow exponentially in the treewidth of the crossing graph. This was done to allow for faster experimentation, given the large number of instances. The instances for which Variant 1a could not finish within T' were considered unfeasible for AG-1a2a and AG-1a2b.

Performance of the exact algorithm. Fig. 9 shows the percentage of instances per graph class that the ILP was able to solve within the given time limit of $T = 15$ minutes (900 seconds). In each chart, the x-axis is the time in seconds and the y-axis (blue line) reports the number of instances solved optimally within the time at the corresponding x coordinate. Fig. 10 shows the optimality gap across the instances for each graph class (a point for each instance), where the instances were sorted by the number of crossings; points of instances with the same gap coincide in the chart.

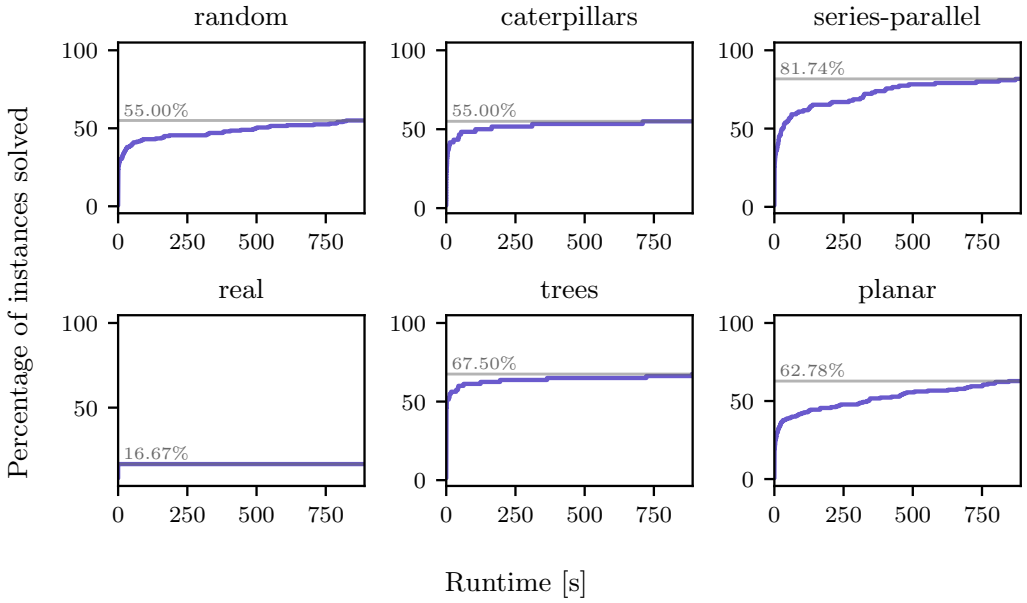


Figure 9 Percentage of instances solved (optimally) by the exact algorithm within time T.

Regarding the Geometric Graphs benchmark, more than half (55%) of the `random graphs` were solved optimally. In particular, all instances with 90 or less crossing edges were solved optimally in the given time, and 90% of the instances with 150 or less crossing edges were solved optimally. Conversely, only the two `real graphs` with small number of crossings (namely, the graphs bmw200 and ca-sandi_auths) could be solved optimally within T. However, for the graph rajat11 (which contains 290 crossings), the optimality gap is close to 0 (namely, 0.08). Overall, for most of the instances in the `random graphs` and `real graphs`, the optimality gap tends to increase with the number of edge crossings.

Regarding the Crossing Graphs benchmark, most of the `series-parallel graphs` (81.74%) and a high percentage of the `trees` (67.5%) and `planar graphs` (62.78%) could be solved optimally. Interestingly, more randomly sampled trees were solved than caterpillars (55.00%) in the given time. However, it is worth noting that the regular pattern observed in the optimality gaps for the `caterpillars` and `trees` is due to the fact that, for instances not solved to optimality within the time limit, the absolute difference between the best bound and the incumbent solution was exactly 1.

Performance of the heuristics. Full details on the runtime of all the heuristics can be found in the long version of the paper [5]. Heuristics AG-1b2a, AG-1b2b, AG-1c2a, and AG-1c2b computed all the instances of our graph benchmark very efficiently. On average they took less than 0.1 seconds on the `random graphs` and about 0.34 seconds on `real graphs` (the most expensive required about 1.6 seconds). Also, they took less than 10 milliseconds for the instances in the Crossing Graphs datasets. Runtime differences among these heuristics are negligible across Variants 2a and 2b for Phase 2, although 2a is slightly faster than 2b. The runtime is mostly affected by the variant chosen for Phase 1, where Variant 1b is a bit faster than Variant 1c, as expected.

Conversely, Variant 1a is significantly more expensive. We implemented it, using the Minimum Fill-in heuristic [8] to compute tree decompositions. For the analysis of the heuristics based on this variant, we took into account only those instances where a Pareto-

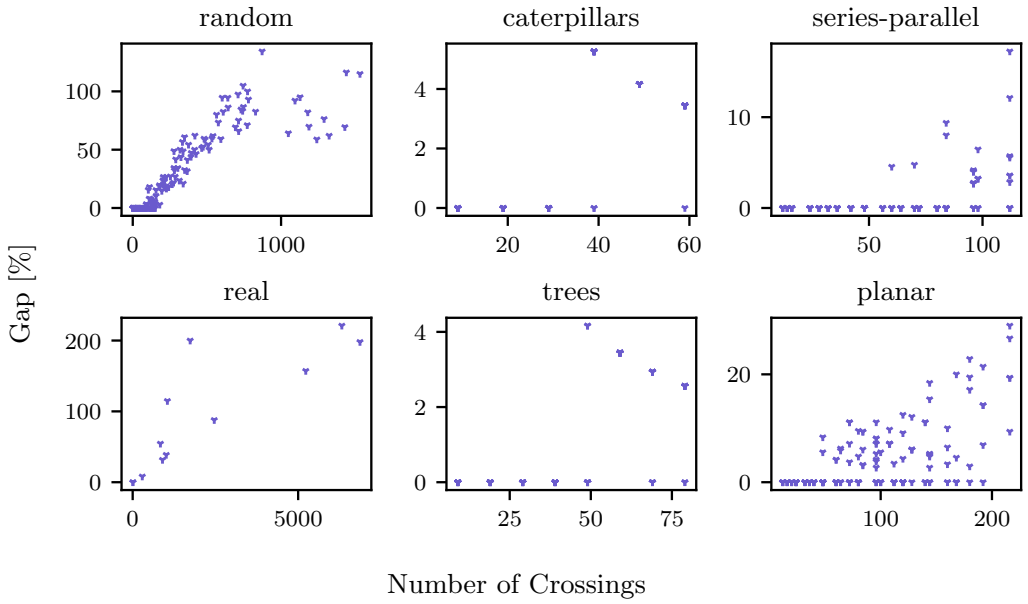


Figure 10 The optimality gap (%) reached by the ILP solver in the given time for each graph class. The instances of each graph class is sorted in ascending order by the number of crossings.

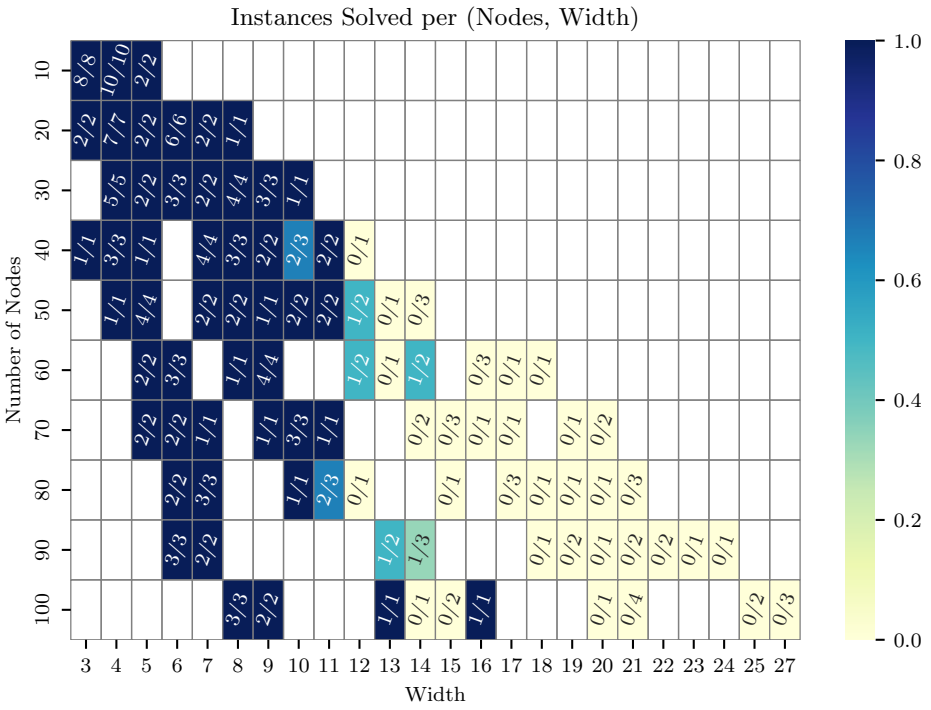


Figure 11 Fraction of the instances in the `random graphs` dataset terminated by Variant `1a` within T', with respect to the number of graph nodes and the width of the tree decomposition.

Table 2 Runtime in seconds for the different heuristics on the `real graphs`. The empty cells refer to the instances that could not be solved by Variant 1a with the given time limit.

graph name	AG-1a2a	AG-1a2b	AG-1b2a	AG-1b2b	AG-1c2a	AG-1c2b
bwm200	0.00	0.00	0.00	0.00	0.00	0.00
ca-sandi_auths	0.00	0.00	0.00	0.00	0.00	0.00
rajat11	0.09	0.13	0.06	0.08	0.10	0.13
lesmis			0.05	0.05	0.08	0.08
ca-netscience			0.61	1.00	1.00	1.00
gd06_theory			0.07	0.07	0.07	0.07
road-chesapeake			0.04	0.04	0.05	0.05
insecta-beetle-group-c1-period-1			0.04	0.04	0.06	0.06
polbooks			0.32	0.32	0.53	0.53
email-enron-only			1.05	1.05	1.64	1.64
eco-stmarks			0.32	0.32	0.40	0.40
adjnoun			0.53	0.53	0.70	0.70

Table 3 The performance of our heuristics in comparison to the exact algorithm.

	All Graphs Combined			
	% Opt	Avg approx. ratio	Min–Max approx. ratio	SD approx. ratio
AG-1a2a	59.9% (243/406)	94.14%	50.00% – 100.00%	9.87%
AG-1a2b	63.1% (256/406)	94.72%	50.00% – 100.00%	9.37%
AG-1b2a	5.7% (23/406)	67.91%	33.33% – 100.00%	13.44%
AG-1b2b	5.4% (22/406)	68.21%	42.86% – 100.00%	13.07%
AG-1c2a	14.3% (58/406)	82.76%	44.44% – 100.00%	10.36%
AG-1c2b	15.3% (62/406)	83.45%	50.00% – 100.00%	10.35%

optimal maximum pair of independent sets was found within the given time $T' = 10$ minutes. Variant 1a was able to find a solution for all the instances in the Crossing Graphs benchmark, as the Minimum Fill-in heuristic produced tree decompositions of width at most six for planar graphs and at most two for series-parallel graphs. For the `random graphs`, Variant 1a solved 137 (of the 200) instances within time T'; the fraction of instances solved for each combination of the two parameters "number of nodes" and "width of the tree decomposition" is illustrated in the heatmap of Fig. 11. For the `real graphs`, Variant 1a solved only 3/12 instances within T' (see Table 2).

Regarding the effectiveness of our heuristics, Table 3 summarizes the comparison over all instances that the exact algorithm managed within the given time $T = 15$ minutes. Detailed results for the different types of graphs are reported in [5].

The results show that AG-1a2a and AG-1a2b achieve a $\sim$ 60% optimality rate and an average approximation ratio of $\sim$ 95%, significantly outperforming the alternatives AG-1b2a/AG-1b2b and AG-1c2a/AG-1c2b in both metrics. The standard deviation of the approximation ratio for AG-1a2a and AG-1a2b remains below 10%, indicating consistent performance. Notably, for the `trees` and the `caterpillars`, AG-1a2a and AG-1a2b achieve near-perfect optimality rates (97.0% and 100.0%, respectively) with approximation ratios close to 100%. In contrast, the heuristics based on Variant 1b achieve fewer than 6% optimal

solutions on average, with lower approximation ratios around 68%. The heuristics using Variant 1c, while also exhibiting low optimality rates ($\sim 15\%$), maintain a stronger average approximation ratio of $\sim 83\%$ and a good standard deviation ($\sim 10.4\%$).

We remark that, for the `real graphs` the exact algorithm managed only 2 instances within the given time T, and AG-1a2a and AG-1a2b managed only 1 additional instance within the given time T'. This data is not enough to get a clear picture of how the other heuristics compare on this dataset. However, by examining the data for all the instances in the `real graphs` dataset, we confirm that also in this case the heuristics based on Variant 1c outperformed those based on Variant 1b, with an improvement of about 29% on average.

Main findings. The exact algorithm typically handles geometric graphs with up to $90 - 100$ crossing edges in few minutes. However, it becomes generally impractical for real-world graphs that are locally dense and inevitably contain many more crossing edges. Regarding the heuristics, the difference between Variants 2a and 2b, in terms of both efficiency and effectiveness, is often negligible. A more detailed analysis of this phenomenon is given in [5]. Conversely, the choice of the variant for Phase 1 makes a big difference, namely:

- The heuristics based on Variant 1a yield solutions close to the optimum in many cases. However, they become computationally unfeasible for graphs with high number of nodes or with crossing graphs of high treewidth; in particular, they exhibit more or less the same limits as the exact algorithm on the real graphs, although they make it possible to manage all instances in the Crossing Graphs benchmark.
- The heuristics based on Variant 1c offer the best trade-off between efficiency and effectiveness. They perform fast on all instances and maintain high average approximation ratios (approximately $80\% - 88\%$, depending on the graph class).

8 Final Remarks and Future Work

We have proposed the MaxMinFramePlanarStory model, where we visualize a graph in a sequence of planar frames such that only one edge at the time enters the story and the minimum frame size is maximized. We suggest the following research directions.

RD1. In addition to ensuring drawing stability and low visual complexity, our model promotes smooth transitions between consecutive frames (i.e., minimal local changes from one frame to the next), as only the current edges that intersect the upcoming edge are removed. However, this approach may result in long stories. To balance these aspects, we can compress a planar story by allowing multiple edges to be inserted simultaneously: for each subsequence of consecutive, pairwise non-crossing edges, all of them can be added at once. In this way, the resulting compressed story remains planar, and its smallest frame is at least as large as that of the original story. Nonetheless, enabling multiple edges to enter simultaneously in the visualization motivates the design of dedicated algorithms aimed at producing short stories while still maximizing the minimum frame size.

RD2. We focused on constructing a planar story of geometric graphs, i.e., of graphs with a given drawing. It would be interesting to investigate how different drawings of the same graph affect the quality of the story. Furthermore, what kind of crossing graphs would allow us to construct good edge stories?

RD3. Our heuristics based on Variants 1b and 1c are fast enough to solve also more complex instances than those in our benchmark. It would be interesting to investigate whether the quality of these variants could be improved by using different heuristics for choosing the independent sets for the initial and final frames than just adding vertices of minimum degree. See, e.g., [20] for a list of heuristics for computing large independent sets.

RD4. Regarding the problem complexity: Is MaxMinFramePlanarStoryD(G,m) NP-complete for 3-plane graphs? Is the problem in FPT parameterized by the treewidth of the crossing graph of G? Is Advanced Greedy 1a a constant-factor approximation algorithm?

References

1 Laurent Alonso, Jean-Luc Rémy, and René Schott. A linear-time algorithm for the generation of trees. *Algorithmica*, 17(2):162–183, 1997. doi:10.1007/BF02522824.

2 Daniel Archambault, Giuseppe Liotta, Martin Nöllenburg, Tommaso Piselli, Alessandra Tappini, and Markus Wallinger. Bundling-aware graph drawing. In Stefan Felsner and Karsten Klein, editors, *32nd International Symposium on Graph Drawing and Network Visualization (GD 2024)*, volume 320 of *LIPIcs*, pages 15:1–15:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.GD.2024.15.

3 Ivan J. Balaban. An optimal algorithm for finding segments intersections. In *Proceedings of the 11th Annual Symposium on Computational Geometry (SoCG'95)*, pages 211–219. Association for Computing Machinery, 1995. doi:10.1145/220279.220302.

4 Fabian Beck, Michael Burch, Stephan Diehl, and Daniel Weiskopf. A taxonomy and survey of dynamic graph visualization. *Comput. Graph. Forum*, 36(1):133–159, 2017. doi:10.1111/CGF.12791.

5 Carla Binucci, Sabine Cornelsen, Walter Didimo, Seok-Hee Hong, Eleni Katsanou, Maurizio Patrignani, Antonios Symvonis, and Samuel Wolf. Planar stories of graph drawings: Algorithms and experiments. Tech. Report arXiv:2508.17532, Cornell University, 2025. doi:10.48550/arXiv.2508.17532.

6 Carla Binucci, Emilio Di Giacomo, William J. Lenhart, Giuseppe Liotta, Fabrizio Montecchiani, Martin Nöllenburg, and Antonios Symvonis. On the complexity of the storyplan problem. In Patrizio Angelini and Reinhard von Hanxleden, editors, *30th International Symposium Graph Drawing and Network Visualization (GD'22)*, volume 13764 of *Lecture Notes in Computer Science*, pages 304–318. Springer, 2022. doi:10.1007/978-3-031-22203-0_22.

7 Carla Binucci, Emilio Di Giacomo, William J. Lenhart, Giuseppe Liotta, Fabrizio Montecchiani, Martin Nöllenburg, and Antonios Symvonis. On the complexity of the storyplan problem. *J. Comput. Syst. Sci.*, 139:103466, 2024. doi:10.1016/J.JCSS.2023.103466.

8 Hans L. Bodlaender. Discovering treewidth. In Peter Vojtáš, Mária Bieliková, Bernadette Charron-Bost, and Ondrej Sýkora, editors, *SOFSEM 2005: Theory and Practice of Computer Science*, pages 1–16, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. doi:10.1007/978-3-540-30577-4_1.

9 Manuel Borrazzo, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, and Maurizio Patrignani. Graph stories in small area. *J. Graph Algorithms Appl.*, 24(3):269–292, 2020. doi:10.7155/JGAA.00530.

10 Jérémie Chalopin and Daniel Gonçalves. Every planar graph is the intersection graph of segments in the plane: extended abstract. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*, pages 631–638. ACM, 2009. doi:10.1145/1536414.1536500.

11 Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W. Klau, Karsten Klein, and Petra Mutzel. The open graph drawing framework (OGDF). In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization*, pages 543–569. Chapman and Hall/CRC, 2013.

12 Giuseppe Di Battista, Walter Didimo, Luca Grilli, Fabrizio Grosso, Giacomo Ortali, Maurizio Patrignani, and Alessandra Tappini. Small point-sets supporting graph stories. In Patrizio Angelini and Reinhard von Hanxleden, editors, *30th International Symposium on Graph Drawing and Network Visualization (GD 2022)*, volume 13764 of *Lecture Notes in Computer Science*, pages 289–303. Springer, 2022. doi:10.1007/978-3-031-22203-0_21.

13 Giuseppe Di Battista, Walter Didimo, Luca Grilli, Fabrizio Grosso, Giacomo Ortali, Maurizio Patrignani, and Alessandra Tappini. Small point-sets supporting graph stories. *J. Graph Algorithms Appl.*, 27(8):651–677, 2023. doi:10.7155/JGAA.00639.

14 Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, Fabrizio Montecchiani, and Ioannis G. Tollis. Exploring complex drawings via edge stratification. In Stephen K. Wismath and Alexander Wolff, editors, *Graph Drawing - 21st International Symposium, GD 2013, Bordeaux, France, September 23-25, 2013, Revised Selected Papers*, volume 8242 of *Lecture Notes in Computer Science*, pages 304–315. Springer, 2013. doi:10.1007/978-3-319-03841-4_27.

15 Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, Fabrizio Montecchiani, and Ioannis G. Tollis. Techniques for edge stratification of complex graph drawings. *J. Vis. Lang. Comput.*, 25(4):533–543, 2014. doi:10.1016/J.JVLC.2014.05.001.

16 Peter Eades, Quan Hoang Nguyen, and Seok-Hee Hong. Drawing big graphs using spectral sparsification. In Fabrizio Frati and Kwan-Liu Ma, editors, *25th International Symposium on Graph Drawing and Network Visualization (GD 2017)*, volume 10692 of *Lecture Notes in Computer Science*, pages 272–286. Springer, 2017. doi:10.1007/978-3-319-73915-1_22.

17 Jirí Fiala, Oksana Firman, Giuseppe Liotta, Alexander Wolff, and Johannes Zink. Outerplanar and forest storyplans. In Henning Fernau, Serge Gaspers, and Ralf Klasing, editors, *49th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2024)*, volume 14519 of *Lecture Notes in Computer Science*, pages 211–225. Springer, 2024. doi:10.1007/978-3-031-52113-3_15.

18 Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991. doi:10.1002/spe.4380211102.

19 Daniel Gonçalves, Lucas Isenmann, and Claire Pennarun. Planar graphs as L-intersection or L-contact graphs. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2018)*, pages 172–184. SIAM, 2018. doi:10.1137/1.9781611975031.12.

20 Ernestine Großmann, Sebastian Lamm, Christian Schulz, and Darren Strash. Finding near-optimal weight independent sets at scale. *Journal of Graph Algorithms and Applications*, 28(1):439–473, 2024. doi:10.7155/jgaa.v28i1.2997.

21 Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. URL: https://www.gurobi.com.

22 Aric A Hagberg, Daniel A Schult, and Pieter J Swart. Exploring network structure, dynamics, and function using networkx. *Proceedings of the 7th Python in Science Conferences (SciPy 2008)*, pages 11–15, 2008.

23 Danny Holten and Jarke J. van Wijk. Force-directed edge bundling for graph visualization. *Comput. Graph. Forum*, 28(3):983–990, 2009. doi:10.1111/J.1467-8659.2009.01450.X.

24 Seok-Hee Hong, Quan Hoang Nguyen, Amyra Meidiana, Jiaxi Li, and Peter Eades. BC tree-based proxy graphs for visualization of big graphs. In *IEEE PacificVis 2018*, pages 11–20, 2018. doi:10.1109/PACIFICVIS.2018.00011.

25 Takehiro Ito, Erik D. Demaine, Nicholas J.A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12):1054–1065, 2011. doi:10.1016/j.tcs.2010.12.005.

26 Takehiro Ito, Marcin Kamiński, Hirotaka Ono, Akira Suzuki, Ryuhei Uehara, and Katsuhisa Yamanaka. Parameterized complexity of independent set reconfiguration problems. *Discrete Applied Mathematics*, 283:336–345, 2020. doi:10.1016/j.dam.2020.01.022.

27 Yosuke Kikuchi, Hiroyuk Tanaka, Shin-ichi Nakano, and Yukio Shibata. How to obtain the complete list of caterpillars. In Tandy Warnow and Binhai Zhu, editors, *Computing and Combinatorics*, pages 329–338, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

28 Stephen G. Kobourov. Force-directed drawing algorithms. In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization*, pages 383–408. Chapman and Hall/CRC, 2013.

29 J. Kratochvíl and J. Matousek. Intersection graphs of segments. *Journal of Combinatorial Theory, Series B*, 62(2):289–315, 1994. doi:10.1006/jctb.1994.1071.

30 Bojan Mohar. Face covers and the genus problem for apex graphs. *Journal of Combinatorial Theory, Series B*, 82(1):102–117, 2001. doi:10.1006/jctb.2000.2026.

31 Quan Hoang Nguyen, Seok-Hee Hong, and Peter Eades. TGI-EB: A new framework for edge bundling integrating topology, geometry and importance. In Marc J. van Kreveld and Bettina Speckmann, editors, *19th International Symposium on Graph Drawing (GD 2011)*, volume 7034 of *Lecture Notes in Computer Science*, pages 123–135. Springer, 2011. doi:10.1007/978-3-642-25878-7_13.

32 Quan Hoang Nguyen, Seok-Hee Hong, Peter Eades, and Amyra Meidiana. Proxy graph: Visual quality metrics of big graph sampling. *IEEE Trans. Vis. Comput. Graph.*, 23(6):1600–1611, 2017. doi:10.1109/TVCG.2017.2674999.

33 Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, 2006. doi:10.1093/ACPROF:OSO/9780198566076.001.0001.

34 Helen C. Purchase. Which aesthetic has the greatest effect on human understanding? In Giuseppe Di Battista, editor, *5th International Symposium on Graph Drawing (GD '97)*, volume 1353 of *Lecture Notes in Computer Science*, pages 248–261. Springer, 1997. doi:10.1007/3-540-63938-1_67.

35 Helen C. Purchase. Effective information visualisation: a study of graph drawing aesthetics and algorithms. *Interact. Comput.*, 13(2):147–162, 2000. doi:10.1016/S0953-5438(00)00032-1.

36 Helen C. Purchase, David A. Carrington, and Jo-Anne Allder. Empirical evaluation of aesthetics-based graph layout. *Empir. Softw. Eng.*, 7(3):233–255, 2002.

37 Helen C. Purchase, Christopher Pilcher, and Beryl Plimmer. Graph drawing aesthetics - created by users, not algorithms. *IEEE Trans. Vis. Comput. Graph.*, 18(1):81–92, 2012. doi:10.1109/TVCG.2010.269.

38 D. Rafiei. Effectively visualizing large networks through sampling. In *VIS 05. IEEE Visualization, 2005.*, pages 375–382, 2005. doi:10.1109/VISUAL.2005.1532819.

39 Neil Robertson and Paul D. Seymour. Graph minors. iii. planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984. doi:10.1016/0095-8956(84)90013-3.

40 Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. URL: https://networkrepository.com.

41 Yanhong Wu, Nan Cao, Daniel Archambault, Qiaomu Shen, Huamin Qu, and Weiwei Cui. Evaluation of graph sampling: A visualization perspective. *IEEE Trans. Vis. Comput. Graph.*, 23(1):401–410, 2017. doi:10.1109/TVCG.2016.2598867.

42 Hong Zhou, Panpan Xu, Xiaoru Yuan, and Huamin Qu. Edge bundling in information visualization. *Tsinghua Science and Technology*, 18(2):145–156, 2013. doi:10.1109/TST.2013.6509098.

Edge Densities of Drawings of Graphs with One Forbidden Cell

Benedikt Hahn
Graz University of Technology, Austria

Torsten Ueckerdt
Karlsruhe Institute of Technology, Germany

Birgit Vogtenhuber
Graz University of Technology, Austria

Abstract

A connected topological drawing of a graph divides the plane into a number of cells. The type of a cell c is the cyclic sequence of crossings and vertices along the boundary walk of c. For example, all triangular cells with three incident crossings and no incident vertex share the same cell type. When a non-homotopic drawing of an n-vertex multigraph G does not contain any such cells, Ackerman and Tardos [JCTA 2007] proved that G has at most $8n - 20$ edges, while Kaufmann, Klemz, Knorr, Reddy, Schröder, and Ueckerdt [GD 2024] showed that this bound is tight.

In this paper, we initiate the in-depth study of non-homotopic drawings that do not contain one fixed cell type $\mathfrak{c}$, and investigate the edge density of the corresponding multigraphs, i.e., the maximum possible number of edges. We consider non-homotopic as well as simple drawings, multigraphs as well as simple graphs, and every possible type of cell. For every combination of drawing style, graph type, and cell type, we give upper and lower bounds on the corresponding edge density. With the exception of the cell type with four incident crossings and no incident vertex, we show for every cell type $\mathfrak{c}$ that the edge density of n-vertex (multi)graphs with $\mathfrak{c}$-free drawings is either quadratic in n or linear in n. In most cases, our bounds are tight up to an additive constant.

Additionally, we improve the current lower bound on the edge density of simple graphs that admit a non-homotopic quasiplanar drawing from $7n - 28$ to $7.5n - 28$.

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

Keywords and phrases Edge density, cell types, forbidden substructures, non-homotopic drawings, simple drawings

Digital Object Identifier 10.4230/LIPIcs.GD.2025.33

Funding *Benedikt Hahn*: This research was funded in whole or in part by the Austrian Science Fund (FWF) 10.55776/DOC183.
Torsten Ueckerdt: funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 520708409.
Birgit Vogtenhuber: This research was funded in whole or in part by the Austrian Science Fund (FWF) 10.55776/DOC183.

1 Introduction

Many fundamental classes of graphs are defined by admitting a drawing without a forbidden configuration or pattern. This includes planar graphs (forbidding two crossing edges), as well as many beyond-planar graph classes such as k-planar graphs (forbidding an edge with $k + 1$ crossings), quasiplanar graphs (forbidding three pairwise crossing edges), fan planar graphs (forbidding an edge crossed by two independent edges[1]) and many others. Planar and

[1] The actual definition is more complex but irrelevant here.


© Benedikt Hahn, Torsten Ueckerdt, and Birgit Vogtenhuber;
licensed under Creative Commons License CC-BY 4.0



beyond-planar graphs are of eminent importance, both from a practical and a theoretical point of view. Nevertheless, many fundamental properties are still not sufficiently well understood. One of the most fundamental questions in this area concerns the *edge density* of a beyond-planar graph class $\mathcal{G}$, that is, the maximum possible number of edges of any n-vertex graph in $\mathcal{G}$. There is a long history of edge density results ranging from k-planar graphs [2, 20, 21], k-quasiplanar graphs [1, 3], k-bend RAC graphs [4, 8], (k, l)-grid free graphs [19], and more to the recently introduced Density Formula [16].

We study the *cells* of a drawing, that is, the regions of $\mathbb{R}^2$ delimited by the vertices and edges of the drawing. The *type* of a cell c is the cyclic order of edge segments, vertices, and crossings along the boundary of c. For example, a  -cell is incident to three edge segments, three crossings, and no vertex. Distinguishing cells by their type is often useful when reasoning about drawings. The recently developed Density Formula [16], whose many applications usually come down to counting and relating different cell types in drawings, has further increased the need for a good grasp on the relation between types of cells in a drawing and the structure of the underlying graph. Interestingly, Ackerman and Tardos [3] proved already in 2007 that n-vertex drawings with no -cell contain at most $8n - 20$ edges.

Our contribution. In this paper, we consider for each cell type $\mathfrak{c}$ the corresponding beyond-planar graph class $\mathcal{G}_\mathfrak{c}$ consisting of all graphs that admit a drawing without $\mathfrak{c}$-cells. We present upper and lower bounds on the edge densities of graphs in $\mathcal{G}_\mathfrak{c}$. We determine the asymptotic edge density in the setting of arbitrary, non-homotopic, and simple drawings, except for one particular cell type. Depending on $\mathfrak{c}$, we obtain some linear and some quadratic edge densities. In many cases, our bounds are tight up to an additive constant. All our results and some previous related work are shown in Table 1. Our lower bounds entail various novel constructions, while our upper bounds are mostly proved via discharging. Additionally, we present a construction of simple graphs on $7.5n - 28$ edges that admit a non-homotopic quasiplanar drawing, beating a previous lower bound by $\frac{n}{2}$ in the highest-order term.

Related work. Apart from the above-mentioned work of Ackerman and Tardos [3] and Kaufmann, Klemz, Knorr, Reddy, Schröder, and Ueckerdt [16] on -free drawings, we are (as far as we know) the first to consider graph drawings with a forbidden type of cell. However, some beyond-planar drawing styles are characterized by forbidding a configuration that is somewhat similar to a particular cell type. For example, in a fan-crossing-free drawing [7] there is no edge crossed by two adjacent edges, which resembles the -cell. Secondly, in a quasiplanar drawing, there are no three pairwise crossing edges, which resembles the -cell. And in a $(2, 2)$-grid-free drawing, there are no two pairs e_1, e_2, f_1, f_2 of edges such that every e_i crosses every f_j, which resembles the -cell. For a collection of edge density results on these and other beyond-planar graph classes, we refer to [9, Chapter 4]. Let us remark that in the three cases above, every fan-crossing-free (quasiplanar, $(2, 2)$-grid-free) drawing is in particular a -free (-free, -free) drawing. Hence, lower bound constructions carry over to our setting, while in general, $\mathfrak{c}$-free drawings might allow for a much higher edge density.

Let us also mention that cell types have been investigated in the context of arrangements of pseudolines. For simple arrangements of pseudolines, it is known that only -cells are forced to appear in every sufficiently large arrangement. In fact, any simple arrangement of n pseudolines in the Euclidean plane or in the projective plane contains at least $n - 2$ or n, respectively, -cells [11, 18]. On the other hand, there are various constructions without -cells [13, 14], and arbitrarily large simple arrangements consisting only of -cells and -cells [17]. Further results on the number of cells of different types in pseudoline

Table 1 Results on edge densities of graphs admitting a drawing without one fixed type of cell. Upper bounds hold for all $n \geq 3$, while lower bounds are proven for infinitely many values of n.

Forbidden cell	Graph type	Drawing type	Lower Bound	Upper Bound
Any	Simple	No restriction	$\binom{n}{2}$ Construction 3	$\binom{n}{2}$
3-cell	Multi	Non-homotopic	$8n - 20$ [16]	$8n - 20$ [3]
	Simple	Non-homotopic	$8n - 28$ Construction 8	$8n - 20$ [3]
	-	Simple	$7n - 30$ Construction 6	$7n - 20$ Theorem 5
4-cell	Multi	Non-homotopic	$9n - 18$ Construction 5	-
	Simple	Non-homotopic	$6n - 12$ Construction 4	$\binom{n}{2}$
	-	Simple	$6n - 12$ Construction 4	$\binom{n}{2}$
4-cell	-	Simple	$\Omega(n^2)$ Construction 9	$\binom{n}{2}$
5-cell	Multi	Non-homotopic	$6n - 12$ Construction 4	$6n - 12$ Theorem 4
	-	Simple	$6n - 12$ Construction 4	$6n - 12$ Theorem 4
Others	-	Simple	$\binom{n}{2}$ Construction 1	$\binom{n}{2}$

arrangements can be found in the textbook chapter [10], including upper bounds on the number of 3-cells and 4-cells in simple pseudoline arrangements from [6, 12], and a lower bound on the number of 4-cells and 5-cells from [22].

Outline. We start in Section 2 with relevant definitions and some statements that we will use throughout the paper. Then, in Section 3 we show that most cell types can be avoided in simple drawings of K_n and that all cell types can be avoided in arbitrary drawings of K_n. Constituting the main body of this work, we address the remaining cell types in Sections 4–7 with quasiplanarity discussed jointly with 3-cells in Section 6.

2 Preliminaries: Drawings, cells, and cell types

Throughout this paper, we consider finite graphs with no loops but possibly parallel edges. To make the (potential) presence of multiedges more explicit, we sometimes use the term *multigraph*. Similarly, when a graph has no parallel edges, it is called a *simple* graph. A drawing Γ of a multigraph $G = (V, E)$ maps vertices to pairwise distinct points and edges to curves connecting the corresponding vertices. For convenience, we consider drawings on the sphere $\mathbb{S}^2$ to avoid a special treatment of the unbounded cell. As customary, we require that no edge passes through a vertex, any two edges have only finitely many points in common, each being a common endpoint or a proper crossing, and that no three edges cross in the same point. A drawing Γ is *connected* if the image of the map Γ is connected in $\mathbb{S}^2$.

A drawing is *simple* if any two edges have at most one point in common, which is either a common endpoint or a crossing. As this rules out parallel edges, only simple graphs admit simple drawings. A *lens* is a region of $\mathbb{S}^2$ bounded by exactly two parts of edges. This could be two parallel edges, two edges crossing twice, or two crossing adjacent edges. We call a lens *empty* if its interior contains neither a crossing nor a vertex. Then a drawing is *non-homotopic* if it does not contain any empty lens.

Consider any fixed connected drawing Γ of some multigraph G. The crossings split each edge into a number of *edge segments*. An *outer edge segment* is incident to a vertex, while an *inner edge segment* starts and ends with a crossing. The *cells* of Γ are the connected components of $\mathbb{S}^2$ after the removal of all vertices and edges. For a cell c, we denote by $v(c)$ the number of vertices, by $e(c)$ the number of edge segments, and by $e_{in}(c)$ the number of inner edge segments on its boundary, counted with multiplicity. The *size* of c is the sum of the former two: $\|c\| := e(c) + v(c)$. We denote by $\mathcal{C}$ the set of all cells, and set $\mathcal{C}_k = \{c \in \mathcal{C} \colon \|c\| = k\}$ and $\mathcal{C}_{\geq k} = \{c \in \mathcal{C} \colon \|c\| \geq k\}$.

Since Γ is assumed to be a connected drawing of G, the boundary of every cell is connected. The *type* of a cell c is the cyclic sequence of incidences with edge segments, crossings, and vertices along the boundary of c. We emphasize that a cell type $\mathfrak{c}$ only determines the form of incidence (edge segment, vertex, or crossing) and not the participating edge, vertex, or crossing of G. In particular, different incidences might be with the same edge segment, crossing, or vertex. We use pictograms with the size of the cell inscribed (such as  or ) to concisely denote cell types of small size. For convenience, let a cell type $\mathfrak{c}$ also denote the number of all $\mathfrak{c}$-cells in a given drawing. If no cell in Γ has type $\mathfrak{c}$, then Γ is called $\mathfrak{c}$-*free*.

Let us conclude the preliminaries with an observation from [16] and two lemmas, which we will need in the course of this paper.

▶ **Observation 1** (Kaufmann, Klemz, Knorr, Reddy, Schröder, and Ueckerdt [16])**.** *Let Γ be any non-homotopic connected drawing of some multigraph G with at least three vertices. Then,*

- $\mathcal{C}_1 = \mathcal{C}_2 = \emptyset$,
- $\mathcal{C}_3$ *is the set of all* *-cells,*
- $\mathcal{C}_4$ *is the set of all* *-cells and all* *-cells, and*
- $\mathcal{C}_5$ *is the set of all*  *-cells,* *-cells, and* *-cells.*

The next two lemmata will help prove upper bounds for -free drawings in Section 4 and -free drawings in Section 6. Lemma 2 below relates the number of vertices in a graph to the sizes of cells in a drawing of the graph. The statement already appears implicitly in various edge density proofs via discharging (e.g., [1, 2, 5]) and as a key ingredient in the proof for the Density Formula [16, Lemma 3.6]. We thus state it without proof[2].

▶ **Lemma 2.** *For any connected drawing Γ of a multigraph G with at least one edge,*

$$|V(G)| - 2 = \sum_{c \in \mathcal{C}} \left(\frac{1}{4} \|c\| - 1 \right).$$

The next lemma gives a bound on the number of cells of certain types in terms of the number of inner edge segments in large cells of non-homotopic drawings. Similar statements have been implicitly used in prior work such as [2, 3, 16].

[2] As one of many ways to prove it, for example, subtracting the Density Formula with $t = 0$ from the Density Formula with $t = 1$ gives the result.

▶ **Lemma 3.** *In any connected non-homotopic drawing of a multigraph with at least three vertices, it holds that*

$$3 \cdot \text{△}_3 + \text{△}_4 \leq \sum_{c \in \mathcal{C}_{\geq 5}} e_{in}(c).$$

Proof. Fix any non-homotopic drawing of a multigraph. Let $\mathcal{C}_\Delta$ be the set of △$_3$-cells and △$_4$-cells, and $\mathcal{S}^{in}_\Delta$ the set of inner edge segments incident to cells in $\mathcal{C}_\Delta$. Note that $3 \cdot \text{△}_3 + \text{△}_4 = |\mathcal{S}^{in}_\Delta|$ because two cells in $\mathcal{C}_\Delta$ that share an inner edge segment would form an empty lens, contradicting non-homotopicity. We define an injective function $g\colon \mathcal{S}^{in}_\Delta \to \mathcal{S}^{in}_{\geq 5}$, where $\mathcal{S}^{in}_{\geq 5}$ is the set of inner edge segments incident to a cell in $\mathcal{C}_{\geq 5}$. Let $s \in \mathcal{S}^{in}_\Delta$ be an arbitrary inner edge segment at $c \in \mathcal{C}_\Delta$, and let $e, f \in E$ be the two edges that cross s on the boundary of c. Imagine a curve γ that starts in c and leaves c by crossing s. Then γ follows the paths of e, closely to e, between e and f, until it crosses an inner edge segment s' and reaches a cell c' that is not a □$_4$-cell. Note that $s = s'$ is possible, namely, if none of the cells incident to s is a □$_4$-cell.

In any case, $c' \notin \mathcal{C}_\Delta$ as otherwise c, c', and the □$_4$-cells traversed by γ form an empty lens. So, by Observation 1, we have $c' \in \mathcal{C}_{\geq 5}$, and we set $g(s) = s'$. Since the □$_4$-cells traversed by γ can be uniquely traced back from s' to s, g is injective and therefore $3 \cdot \text{△}_3 + \text{△}_4 \leq \sum_{c \in \mathcal{C}_{\geq 5}} e_{in}(c)$, as desired. ◀

3 Drawings of complete graphs

First, we show that for most cell types $\mathfrak{c}$, complete graphs admit simple $\mathfrak{c}$-free drawings.

▶ **Construction 1.** *For every $n \geq 4$, the complete graph K_n admits a simple ⬠$_5$-free drawing in which each cell c fulfills $\|c\| \leq 5$ or $\|c\| = 2n$.*

Proof. Consider the simple drawing of K_n depicted in Figure 1, in which the vertices lie evenly spaced on a horizontal line and each edge follows the shape of a $\wedge$ with a right-angled peak. For $n \geq 4$, all cells in this drawing except the outer cell are △$_3$-, □$_4$-, △$_4$-, ⬠$_5$-, or ⬠$_5$-cells. Clearly, the outer cell c has size $\|c\| = 2n$. ◀

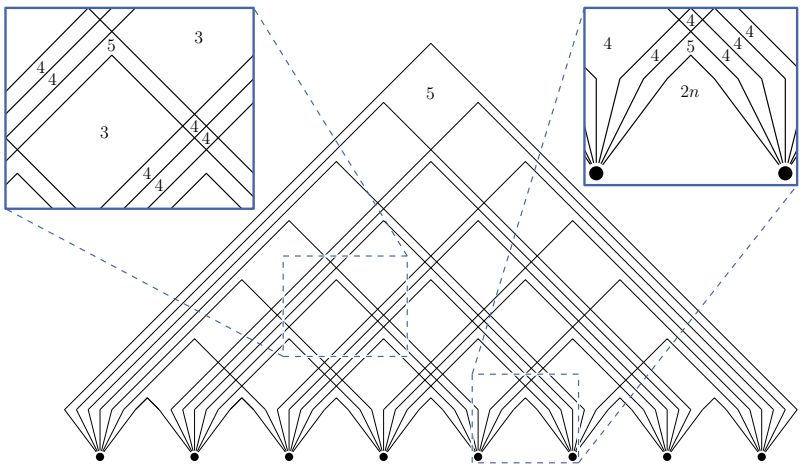


Figure 1 A simple drawing of K_n (here $n = 8$) containing only cells of size at most 5, and one cell of size $2n$.

Construction 1 shows that for any cell type $\mathfrak{c} \notin \{$  , ,  , $\}$, every large enough complete graph admits a simple $\mathfrak{c}$-free drawing. In particular, the edge density of simple $\mathfrak{c}$-free drawings is $\binom{n}{2}$ whenever $\mathfrak{c}$ has size at least 6, or $\mathfrak{c} =$  .

Next, we show that -cells can also be avoided in simple drawings of K_n. Note that any drawing without uncrossed edges will do, see for example [15]. For completeness, we present our own construction.

▶ **Construction 2.** *For every $n \geq 8$, the complete graph K_n has a simple*  *-free drawing.*

Proof. Let $n \geq 8$ be even and start with a straight-line drawing of K_n with vertices in convex position. Then redraw the edges of length 3 as depicted in Figure 2, so as to cross the edges of length 1 (on the convex hull). Further redraw every other edge of length 2 to be uncrossed in the outer cell. Finally, redraw all edges of length $n/2$ to be pairwise crossing in the outer cell. For n odd, take the construction for $n-1$ even, add the last vertex in the central cell, and connect it to all other vertices with straight-line segments. ◀

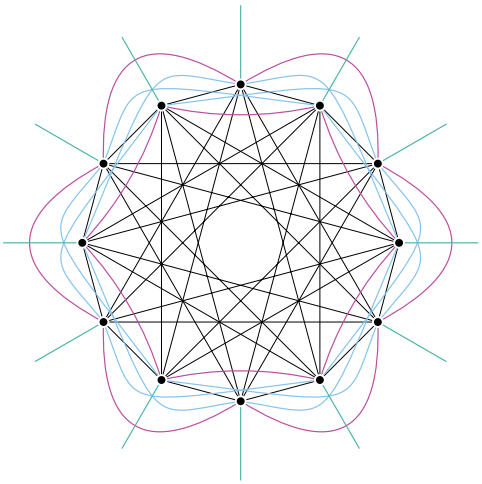


Figure 2 A simple  -free drawing of K_{12}. (A small perturbation eliminates the multi-crossings.) Edges of length three are drawn in blue and those of length two in red. Edges of length $n/2$ are depicted with green edges.

With Constructions 1 and 2 at hand, only the cell types  ,  ,  , and  remain, if we restrict to simple drawings of complete graphs. Before turning to these types, let us close the section by dropping the requirement that the drawing is simple. In fact, without any restriction, every cell type can be avoided in drawings of complete graphs.

▶ **Construction 3.** *For any cell type $\mathfrak{c}$ and infinitely many values of n, the complete graph K_n admits a $\mathfrak{c}$-free drawing.*

Proof. For $\mathfrak{c} \notin \{$, , , $\}$, the claim follows from Constructions 1 and 2. Every remaining cell type $\mathfrak{c}$ has size 3, 4, or 5 and at least one crossing on its boundary. For every $n \in \mathbb{N}$, we now construct a drawing of K_n in which every cell with an incident crossing has size 2 or at least 6, which completes the proof. To do so, start with any drawing of K_n. Locally replace each crossing with the pattern in Figure 3. In the resulting drawing, each cell incident to a crossing is part of such a pattern and therefore of size 2 or at least 6. ◀

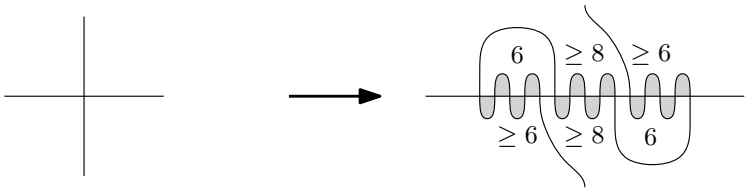


Figure 3 A crossing can be redrawn to bound only cells of size 2 (shaded gray) or at least 6.

4 Non-homotopic drawings without 5-cells

In this section, we consider the 5-cell. In particular, we show that non-homotopic 5-free drawings have only linear edge density. Note that by Construction 3, the restriction to non-homotopic drawings is essential.

▶ **Theorem 4.** *For $n \geq 3$, any connected n-vertex multigraph that admits a non-homotopic 5-free drawing has at most $6n - 12$ edges.*

Proof. Fix a 5-free drawing of an n-vertex multigraph with $n \geq 3$. First, let us compute for each cell $c \in \mathcal{C}$ the quantity $3e(c) + 2v(c) - 12$, and in case $c \in \mathcal{C}_{\geq 5}$ compare it with $e_{in}(c)$.

c is a 3-cell :	$3e(c) + 2v(c) - 12 = -3$
c is a 4-cell :	$3e(c) + 2v(c) - 12 = -1$
c is a 4-cell :	$3e(c) + 2v(c) - 12 = 0$
c is a 5-cell :	$3e(c) + 2v(c) - 12 = 2 = e_{in}(c)$
c is a 5-cell :	$3e(c) + 2v(c) - 12 = 1 \geq 0 = e_{in}(c)$
$\|c\| \geq 6$:	$3e(c) + 2v(c) - 12 = e(c) + 2\|c\| - 12 \geq e(c) \geq e_{in}(c)$

Note that the above case distinction is complete as c is not a 5-cell.

Summing over all cells, this gives

$$\sum_{c \in \mathcal{C}} (3e(c) + 2v(c) - 12) \geq -3 \cdot \text{3} - \text{4} + \sum_{c \in \mathcal{C}_{\geq 5}} e_{in}(c) \geq 0, \tag{1}$$

where the last inequality is an application of Lemma 3.

Finally, we obtain the desired result by double-counting all vertex-cell incidences as

$$2|E| = \sum_{c \in \mathcal{C}} v(c) \overset{(1)}{\leq} \sum_{c \in \mathcal{C}} (3e(c) + 3v(c) - 12) = 3 \sum_{c \in \mathcal{C}} (\|c\| - 4) = 12(n-2),$$

where the last equality is an application of Lemma 2. ◀

Theorem 4 gives an upper bound of $6n - 12$ on the edge density of 5-free non-homotopic multigraphs. As the next construction shows, this bound is tight, even for simple drawings.

▶ **Construction 4.** *For infinitely many values of n, there is a simple n-vertex graph with $6n - 12$ edges that admits a simple 4-free and 5-free drawing.*

Proof. Start with a plane drawing Γ_0 of a 5-connected triangulation G_0 on n vertices. For an edge $e = uv$ in G_0, consider the two triangular faces f, f' in Γ_0 incident to e, and let w, w' be the third vertex (different from u, v) in f, f', respectively. We insert a new edge between w and w' in the drawing by only traversing the two incident faces f, f' of e. As G_0

is 5-connected, ww' is not already an edge in G_0, as otherwise there is a separating triangle in G_0 with w, w' and one of u, v. Now, we do the same for all edges in G_0. Again, as G_0 is 5-connected, for every edge $\tilde{e} \neq e$ in G_0 we obtain a pair different from w, w', as otherwise there is a separating 4-cycle in G_0 with w, w' and one endpoint of each of $e, \tilde{e}$.

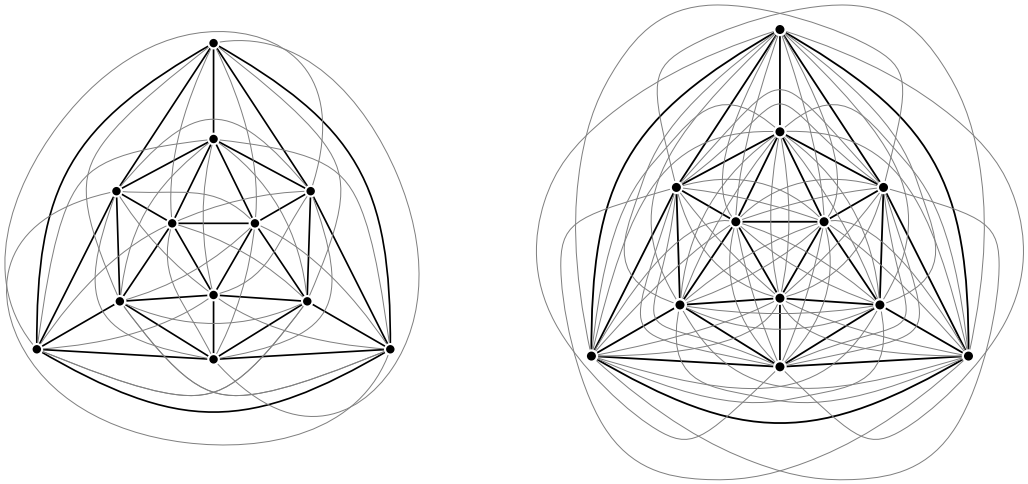


Figure 4 Left: A simple 4-free and 5-free drawing with $6n - 12$ edges. Right: A non-homotopic 4-free drawing of a multigraph with $9n - 18$ edges.

Call the resulting graph G and the resulting drawing Γ. By the above reasoning, the graph G is simple (has no parallel edges). Moreover, observe that the drawing Γ is simple (any two edges have at most one point in common). See Figure 4 (Left) for an illustration.

The only cells in Γ that are not incident to any vertex are 3-cells. Hence, Γ is 4-free and 5-free. Finally, since G_0 is a plane triangulation, we have $|E(G_0)| = 3n - 6$ and thus $|E(G)| = 2|E(G_0)| = 6n - 12$. ◀

5 Non-homotopic drawings without 4-cells

In this section, we briefly discuss the 4-cell. From Construction 4, we immediately obtain a $6n - 12$ lower bound for the edge density of simple graphs that admit a 4-free drawing. A slight variation of this construction yields a better bound for non-homotopic drawings.

▶ **Construction 5.** *For infinitely many values of n, there is an n-vertex multigraph with $9n - 18$ edges that admits a non-homotopic 4-free drawing.*

Proof. As in Construction 4, we start with any 5-connected triangulation G_0 on n vertices. For each edge $e = uv$ in G_0, we again consider the unique pair w, w' of common neighbors of u and v. But this time, we add two parallel edges between w and w', again drawing these new edges so that they only cross the edge e of G_0. For each triangle uvw of G_0, this yields two edges per vertex that emanate from the vertices into the triangle.

Additionally, we draw the new edges in a way that in every triangular face uvw of G_0, the two parallel edges from each vertex enclose exactly two crossings of the four other edges that emanate into that face. This way, uvw is split into one cell of size 6 with no incident vertex and six 3-cells in the interior of uvw, as well as one 5-cell at each edge of uvw and three cells at each vertex of uvw (one 5-cell and two 4-cells); see Figure 4 (right).

Thus, the obtained drawing is $\boxplus$-free. Non-homotopicity is apparent since only the pairs of parallel edges form lenses, which are non-empty. Finally, the drawing contains $9n - 12$ edges, since for each of the $3n - 6$ edges of G_0, two edges are added. ◀

By Construction 5, the edge density of $\boxplus$-free non-homotopic multigraphs is at least $9n - 18$. For simple $\boxplus$-free graphs, it is at least $6n - 12$ by Construction 4. However, we have no non-trivial upper bound; see also Problem 1 in Section 8.

6 Non-homotopic $\triangledown$-free and non-homotopic quasiplanar drawings

As mentioned already, $\triangledown$-free drawings are closely related to *quasiplanar* drawings, that is, drawings in which no three edges cross pairwise. As self-intersection of edges is forbidden, any quasiplanar drawing is $\triangledown$-free but there exist $\triangledown$-free drawings that are not quasiplanar.

Several results from the existing literature contain upper and lower bounds on the edge density of $\triangledown$-free and quasiplanar drawings. See Table 2 for an overview. In fact, the edge density for both drawing styles is known to be $8n - 20$ in the case of non-homotopic drawings of multigraphs. However, there are still some gaps in the case of non-homotopic drawings of simple graphs, and the case of simple drawings of (simple) graphs. In this section, we provide several improvements.

For quasiplanar drawings, we improve the lower bound for non-homotopic drawings of simple graphs from $7n - 28$ [16] to $7.5n - 28$ (Construction 7). We remark that there still remains a gap to the upper bound of $8n - 20$ [3].

For $\triangledown$-free drawings, we improve the lower bound for non-homotopic drawings of simple graphs from $7.5n - \mathcal{O}(1)$ [3] to $8n - 28$ (Construction 8), almost matching the upper bound of $8n - 20$ [3]. For $\triangledown$-free simple drawings of graphs, we give a lower bound of $7n - 30$ (Construction 6) and prove a nearly tight upper bound of $7n - 20$ (Theorem 5). The previous upper bound was $8n - 20$ [3] in this case, too. See again Table 2.

Table 2 Previous and new upper and lower bounds on the number of edges in n-vertex quasiplanar drawings and n-vertex $\triangledown$-free drawings. Upper bounds hold for all $n \geq 3$, while lower bounds are proven for infinitely many values of n.

	Drawing Type	Lower Bounds	Upper Bounds
quasiplanar	Non-homotopic multigraph	$8n - 20$ [16]	$8n - 20$ [3]
	Non-homotopic simple graph	$7n - 28$ implicitly in [16]; $7.5n - 28$ Construction 7	$8n - 20$ [3]
	Simple drawing	$6.5n - 20$ [16]	$6.5n - 20$ [3]
$\triangledown$-free	Non-homotopic multigraph	$8n - 20$ [16]	$8n - 20$ [3]
	Non-homotopic simple graph	$7.5n - \mathcal{O}(1)$ implicitly in [3]; $8n - 28$ Construction 8	$8n - 20$ [3]
	Simple drawing	$7n - 30$ Construction 6 based on [3]	$7n - 20$ Theorem 5; $8n - 20$ [3]

6.1 Upper bound for K_3-free drawings of simple graphs

Ackerman and Tardos proved in [3] that any n-vertex quasiplanar drawing ($n \geq 3$) contains at most $8n - 20$ edges, but explicitly mention that they only use K_3-freeness. For arbitrary (not necessarily non-homotopic) drawings, this would contradict Construction 3. However, their proof relies on the additional assumption that the drawing is crossing-minimal quasiplanar, which they use to rule out cells c with $\|c\| \leq 2$. In arbitrary crossing-minimal K_3-free drawings, this cannot be concluded. But recall from Observation 1 that in any non-homotopic drawing every cell has size at least 3. Therefore, the proof of the $8n - 20$ upper bound in [3] applies to non-homotopic K_3-free drawings, even of multigraphs.

Ackerman and Tardos also considered the case of simple quasiplanar drawings, for which they obtained an upper bound of $6.5n - 20$ edges via a similar proof idea. The final case distinction in this proof however, works only for quasiplanar drawings and not for K_3-free drawings. Here, we extend their proof to simple K_3-free drawings by adding one further case. This yields the slightly higher upper bound of $7n - 20$. In fact, our bound is best possible (up to an additive constant), as we shall later construct simple n-vertex graphs with K_3-free drawings and $7n - 30$ edges (Construction 6). In particular, the edge densities of simple quasiplanar drawings and simple K_3-free drawings are indeed different.

As mentioned above, we modify the proof of Ackerman and Tardos. Hence, we only sketch this proof and refer to [3] for more details.

▶ **Theorem 5.** *Let Γ be a K_3-free drawings of an n-vertex multigraph G with $n \geq 3$. If Γ is non-homotopic, then G has at most $8n - 20$ edges. If Γ is simple, then G has at most $7n - 20$ edges.*

Proof sketch. Consider any K_3-free drawing Γ of an n-vertex multigraph. We later distinguish whether Γ is non-homotopic or simple.

Initially, give each cell charge $\mathrm{ch}(c) := \|c\| - 4$. By Lemma 2, the total charge is therefore $4n - 8$. We will redistribute the charges in two steps and, via double counting, obtain an upper bound on the number of edges. In the first discharging step, we obtain new charges ch' by setting $\mathrm{ch}'(c) := \mathrm{ch}(c) + \frac{1}{5}$ for each 4-cell c and $\mathrm{ch}'(c) := \mathrm{ch}(c) - \frac{e_{in}(c)}{5}$ for each $c \in \mathcal{C}_{\geq 5}$. By Lemma 3, the total charge does not increase in this step. Using K_3-freeness, it can be checked via a case distinction that for every cell c, we have $\mathrm{ch}'(c) \geq \frac{v(c)}{5}$.

The final charge distribution ch'' is obtained by moving charge from cells to vertices. Namely, we distribute the excess charge $\mathrm{ch}'(c) - \frac{v(c)}{5}$ of each cell fairly among its $v(c)$ incident vertices. We lower bound the charge $\mathrm{ch}''(v)$ of an arbitrary vertex v by a case distinction on the type of the cell c_v that is obtained when removing v and its incident edge segments (see Figure 5, left).

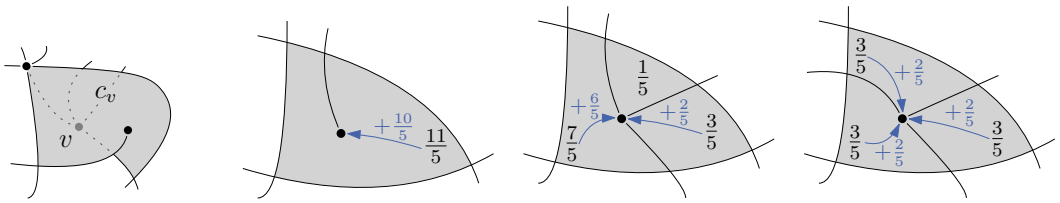


Figure 5 Left: Removing a vertex v and its incident edge segments, a new cell c_v (shaded gray) is formed. Others: Case distinction in Theorem 5 where c_v is a K_3-cell.

In the non-homotopic setting, Ackerman and Tardos prove $\mathrm{ch}''(v) \geq \frac{4}{5}$. The minimum is obtained if c_v is a 2-cell. When Γ is simple quasiplanar, they show $\mathrm{ch}''(v) \geq \frac{7}{5}$ and equality can be obtained when c_v is a 4-cell. To complete the case distinction in our setting of

simple 3-free Γ, it suffices to consider the additional case that c_v is a 3-cell: Note that $\mathrm{ch}''(v)$ is unchanged, whether an edge segment of c_v is crossed once or more than once, since additional crossings only increase the number of 4-cells incident to v, which cannot send any charge to v. Thus, we only have to compute $\mathrm{ch}''(v)$ if $s = 1, 2, 3$ edge segments of c_v are crossed (for $s = 0$, Γ is disconnected), see Figure 5, second-to-left to right. If $s = 1$, v has final charge $\frac{10}{5}$, sent from a cell of size 7 with one incident vertex. If $s = 2$, $\mathrm{ch}''(v) = \frac{6}{5} + \frac{2}{5}$ given by a cell of size 6 with one incident vertex and a 5-cell. Finally, if $s = 3$, $\mathrm{ch}''(v) = 3 \cdot \frac{2}{5}$ from the three incident 5-cells. Thus in total, $\mathrm{ch}''(v) \geq \frac{6}{5}$ for every vertex v. Finally, by rearranging

$$4n - 8 = \sum_{c \in C} \mathrm{ch}(c) \geq \sum_{c \in C} \frac{v(c)}{5} + \sum_{v \in V} \mathrm{ch}''(v) \geq \frac{2}{5}|E| + |V| \cdot \min_v \mathrm{ch}''(v),$$

we obtain $|E| \leq 8n - 20$ for non-homotopic and $|E| \leq 7n - 20$ for simple 3-free drawings. ◀

6.2 Lower bounds for 3-free and quasiplanar drawings of simple graphs

Let us start with simple 3-free drawings. We give a lower bound construction that is tight up to an additive constant. In fact, in this case, it is enough to take a subdrawing of the construction in [3, Theorem 4].

▶ **Construction 6.** *For infinitely many values of n, there is a simple n-vertex graph with $7n - 30$ edges that admits a 3-free drawing.*

Proof. Fix $m \geq 1$ and start with an $m \times 3$-grid of hexagons on $n = 8m + 6$ vertices, each hexagon containing a straight-line drawing of $K_6 \setminus e$ where the vertical diagonal is missing (see Figure 6, left). Now, identify opposite vertices along the side of the grid of length m to obtain the surface of a cylinder whose top and bottom boundary contain 6 vertices. In the resulting drawing, every vertex not on the boundary has degree 11. Now, add long edges as depicted in the center of Figure 6 whenever this is possible. This increases the degree of every vertex sufficiently far away from the boundary to 14. After adding three uncrossed edges each on the top and bottom (see Figure 6, right), we arrive at the final drawing Γ. Counting vertex degrees near the boundary, it can be confirmed that the number of edges in Γ is $7n - 30$.

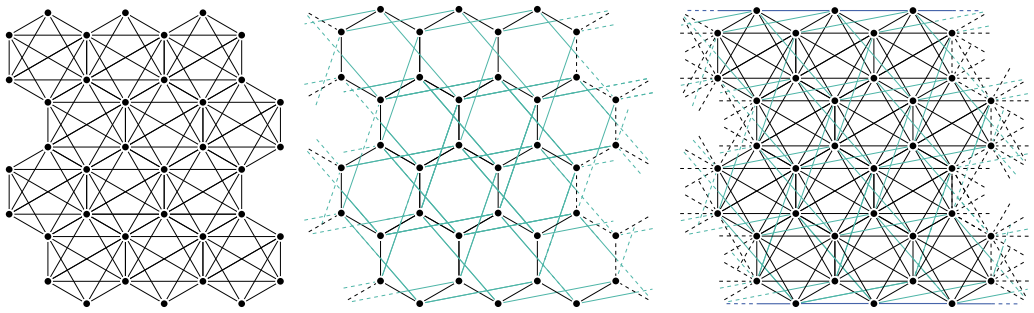


Figure 6 Illustrations of Construction 6.

Before adding the long edges, the drawing is clearly 3-free and no 3-cells are created by adding the long edges and the six planar edges. Finally, the simplicity of Γ follows from the fact that Γ is locally homeomorphic to a straight-line drawing and no two edges are long enough to meet twice on the cylinder. ◀

In the remainder of this section, we construct non-homotopic drawings of simple graphs. We start with a quasiplanar construction, followed by a construction of $\overline{3}$-free such drawings.

▶ **Construction 7.** *For infinitely many values of n, there is a simple n-vertex graph with $7.5n - 28$ edges that admits a non-homotopic quasiplanar drawing.*

Proof. Let $n \geq 14$ be even. We describe a drawing on n vertices with the desired properties. Precise edge paths are obtained by avoiding unnecessary crossings. Figure 7 (Top) depicts the final drawing Γ. We draw

1. an $\frac{n}{2}$-cycle C_{in} as a circle with equidistant points.
2. a concentric copy C_{out} of C_{in} with larger diameter.

We call two radially aligned vertices on C_{in} and C_{out} *partners*. For any vertex x, we denote by x' its partner, by x_{+k} the vertex reached by k clockwise steps on the cycle of x, and by x'_{+k} the partner of x_{+k}. We continue drawing

3. a straight edge between any pair of partners.
4. for any vertex v, a straight edge to v'_{+1}.
5. for any vertex v on C_{in} an edge from v to v_{+2}, drawn through the interior of C_{in}.
6. for any vertex v on C_{out} an edge from v to v_{+2}, drawn through the exterior of C_{out}.
7. for any vertex v on C_{in}, an edge to v'_{+2}. The edge passes through the space between C_{in} and C_{out}, crosses C_{out} in the edge $v'v'_{+1}$ and continues through the exterior of C_{out}.
8. for any vertex v on C_{out}, an edge to v'_{+2}. Edges are drawn symmetrically to the last step.
9. for any vertex v on C_{in} an edge to v_{+3}. The edge first passes through the space between C_{in} and C_{out}, crosses C_{in} between v_{+1} and v_{+2} and then goes through the interior of C_{in}.
10. an edge between v and v_{+3} for any vertex v on C_{out}, symmetrically to the last step.
11. a zig-zag path inside of C_{in} whose edges connect vertices of distance at least 4 on C_{in}.
12. another different zig-zag path in the interior of C_{in} such that no parallel edges are created.
13. a zig-zag path outside of C_{out}, again connecting vertices of distance at least 4 on C_{out}.
14. another such zig-zag path in the exterior of C_{out}.

The first three steps add $\frac{n}{2}$ edges each, step 4 adds n edges, the six steps thereafter $\frac{n}{2}$ edges each, and the last four steps $\frac{n}{2} - 7$ edges each, totaling the desired $7.5n - 28$ edges. As the various steps connect vertices at different distances, the underlying graph is simple.

We now argue that Γ is quasiplanar. First, consider all edges except those in zig-zag paths. These can be locally 3-colored such that one color class consists only of the edges in step 3 and edges of the same color do not cross (see Figure 7). Therefore, any 3 pairwise crossing edges would contain an edge from step 3, but each such edge is only crossed by two other edges, which in turn do not cross.

Now, consider without loss of generality the zig-zag paths P and P' in the interior of C_{in} and some edge $e = uv$ in P. Observe that e is crossed by three sets of edges: (i) Edges in P', (ii) edges from steps 1-10 that cross e at u, and (iii) edges from steps 1-10 that cross e at v.

We argue that no edges from (i), (ii), or (iii) cross. Note that no pair of edges in the same group crosses. Next, an edge from (ii) cannot cross an edge from (iii) since u and v have distance at least 4 on C_{in}, which two edges cannot span while crossing. Further, note that no edge in (i) is incident to u or v. Since edges in (ii) and (iii) cross edges from zig-zag paths only at u and v, they cannot cross edges from (i). Thus, e is not part of three pairwise crossing edges, and in total, the drawing is quasiplanar.

Finally, we argue that Γ is non-homotopic. Note that in the above iterative construction, simplicity is only violated by the edges added in steps 9 and 10. Each such edge is part of four lenses (see dashed edges in Figure 7 (Top)), all of which are non-empty. ◀

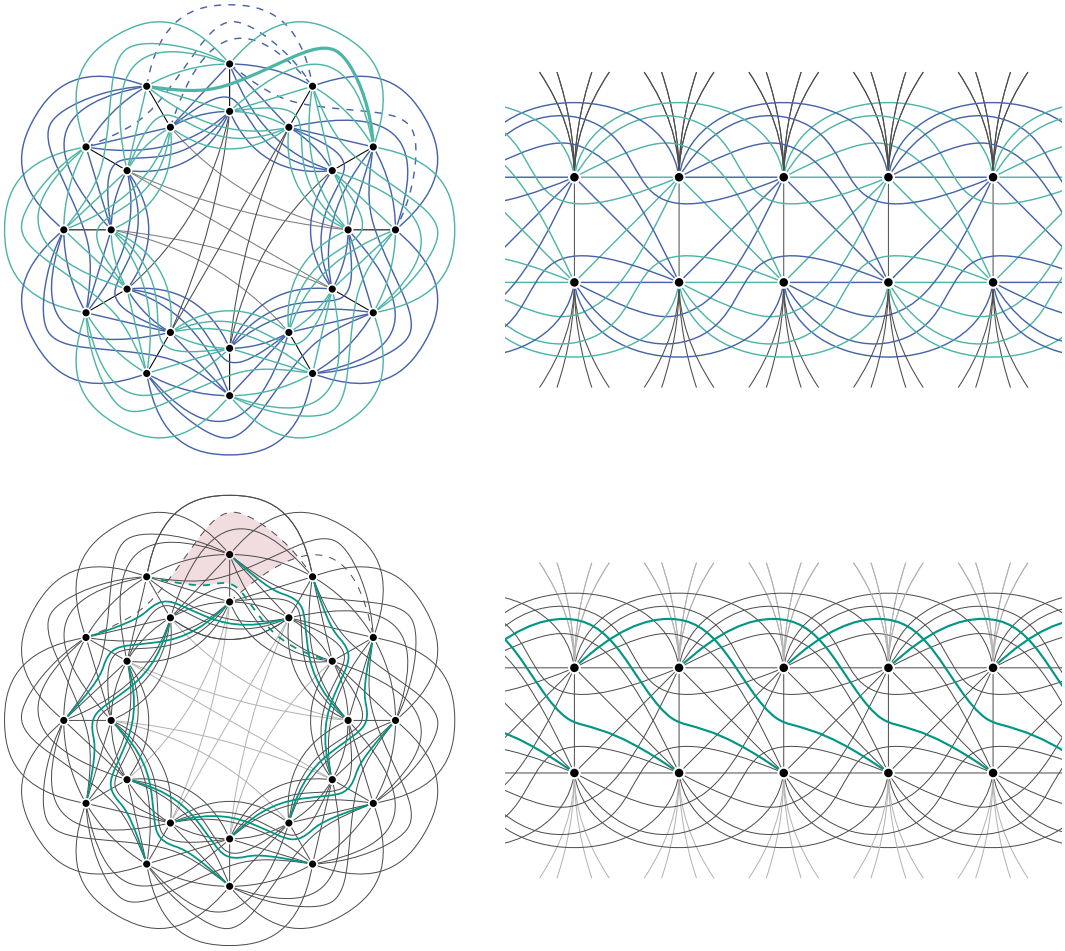


Figure 7 The drawings of Constructions 7 and 8 with the outer zig-zag paths omitted for visual clarity, as well as zoomed-in strips of the local edge patterns.
Top: Construction 7 where edges except on the inner zig-zag paths are partitioned by color into three plane drawings. The edges that form lenses with the thick edge are dashed.
Bottom: Construction 8 where green edges are those not present in Construction 7. The shaded area is bounded by three pairwise crossing edges.

▶ **Construction 8.** *For infinitely many values of n, there is a simple graph on n vertices and $8n - 28$ edges that admits a non-homotopic drawing without $\overset{3}{\triangledown}$-cells.*

Proof. Let $n \geq 14$ be even and start with the $\overset{3}{\triangledown}$-free drawing from Construction 7. We will reuse the notation established there. For any vertex v on C_{out}, we draw an edge from v to v'_{+3} that first moves through the space between C_{in} and C_{out}, crosses C_{in} in $v'_{+1}v'_{+2}$ and then passes through the interior of C_{in}. The resulting drawing is shown in Figure 7 (Bottom).

Checking along the paths of the newly added edges, it can be verified that the drawing remains $\overset{3}{\triangledown}$-free. Note that this process adds $\frac{n}{2}$ edges, none of which are parallel to each other or previous edges. Therefore, we obtain the claimed density. ◀

7 Non-homotopic drawings without  -cells

In this section, we consider the  -cell. In particular, we show that n-vertex graphs with simple  -free drawings can have $\Theta(n^2)$ many edges and minimum degree $\Omega(n)$. The underlying graph in our construction is disconnected but can be made connected by adding a constant number of edges and vertices.

▶ **Construction 9.** *For infinitely many values of n, there is a simple n-vertex graph on $\frac{n^2}{18} - \mathcal{O}(n)$ edges admitting a simple  -free drawing.*

Proof. For each integer $k \geq 2$, we construct a simple k-regular graph $G(k)$ on $n = 3 \cdot (3k-1)$ vertices together with a simple  -free drawing; see Figures 8 and 9 for an illustration. The graph $G(k)$ consists of three isomorphic copies of a graph $G'(k)$ on $3k-1$ vertices. To describe $G'(k)$ (and its drawing), we start with a set A of $3k-1$ equidistant points on a circle C_{out}. The points in A will be the vertices of $G'(k)$, which we denote $a_0, \ldots, a_{3k-2}$ in cyclic ordering around C_{out}. For each $i = 0, \ldots, 3k-2$ draw a straight-line edge e_i between a_i and a_{i+k} (throughout this construction all indices are taken modulo $3k-1$). Let b_i be the crossing point of e_i and e_{i+1}, and B the set of all such b_i.

Then, $b_0, \ldots, b_{3k-2}$ are equidistant points (blue disks in Figure 8) on a smaller circle C_{in} concentric to C_{out}. Note that e_i crosses e_{i+1} at b_i and e_{i-1} at b_{i-1}. Next, we add further edges such that each a_i has exactly k incident edges and at each b_i, exactly k edges pairwise cross. To this end, we insert edges from a_i to $a_{i+k+1}, \ldots, a_{i+\lfloor 3k/2 \rfloor}$ where the edge $a_i a_{i+k+j}$ with $j = 1, \ldots, \lfloor k/2 \rfloor$ goes through b_{i-j-1} and b_{i+2j}; see again Figure 8 for an illustration. This completes the construction of $G'(k)$. Observe that the described drawing of $G'(k)$ is simple (in fact, almost straight-line) and every vertex of $G'(k)$ lies on the same cell.

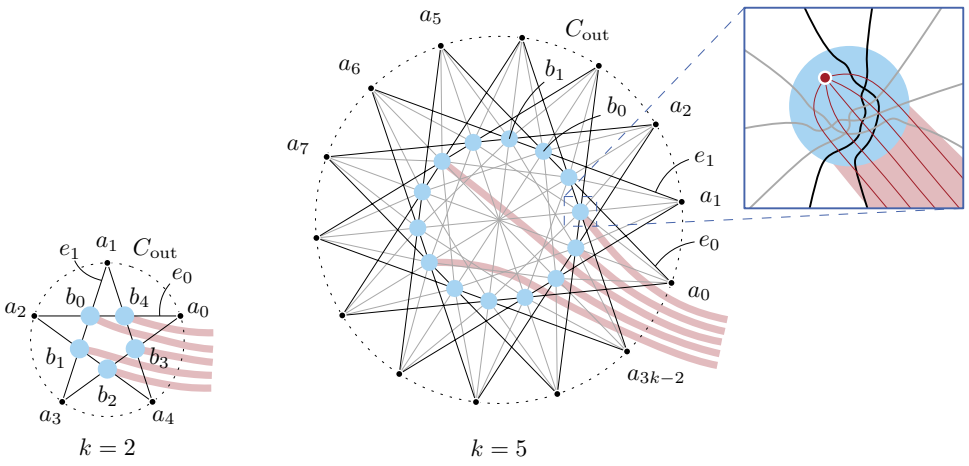


Figure 8 Illustration of the k-regular graph $G'(k)$ on $3k-1$ vertices for $k = 2$ and $k = 5$. At each of the $3k-1$ points in B (blue disks), k edges are pairwise crossing. Corridors are indicated in red. (Due to space reasons, only some corridors are shown for $k = 5$.)

We now take three copies of $G'(k)$, called $G'_1(k), G'_2(k), G'_3(k)$, and interleave the three drawings so as to contain no  -cell. See Figure 9 for an illustration. We shall move the vertices in such a way that each vertex a of $G'_1(k)$ lies close to a point b in the set B of $G'_2(k)$. We locally perturb the k edges of $G'_2(k)$ that all cross in b so that each of the k edges of $G'_1(k)$ emanating from a has its first crossing with a different edge of $G'_2(k)$. Loosely speaking, the edges of $G'_2(k)$ at point b "support" the edges of $G'_1(k)$ starting at point a. Similarly, we let the edges of $G'_3(k)$ support the edges of $G'_2(k)$, and the edges of $G'_1(k)$ support the edges of $G'_3(k)$.

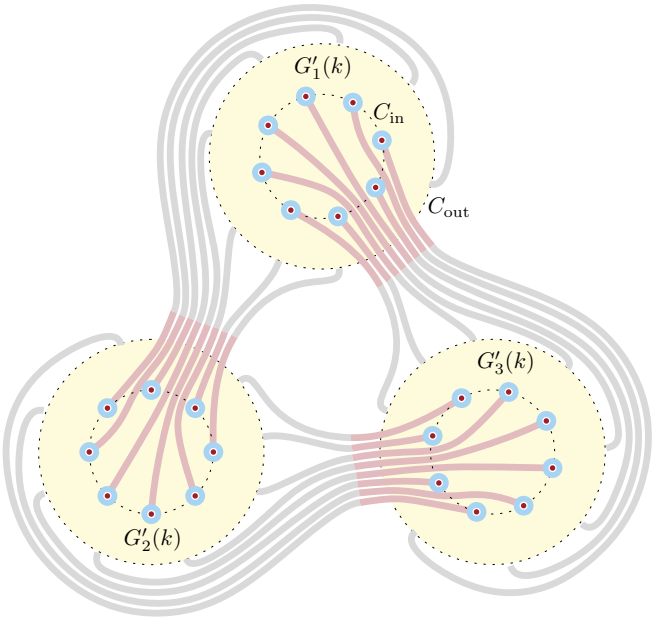


Figure 9 Connecting the drawings of $G'_1(k)$, $G'_2(k)$, and $G'_3(k)$ to a $\mathfrak{c}$-free drawing of $G(k)$.

More precisely, we consider in the drawing of $G'(k)$ for each $b \in B$ a straight and very narrow corridor starting at b and ending on C_{out} between a_0 and a_{3k-2}. Then, we perturb the k edges close to b to obtain a simple drawing in which b sees a segment of each of these k edges. Now, place the simple drawings of $G'_1(k), G'_2(k), G'_3(k)$ with corridors disjointly next to each other. Move the vertices of $G'_1(k)$ (without introducing new crossings) through the prepared corridors of $G'_2(k)$ (each vertex through one corridor). Place each vertex a of $G'_1(k)$ at the end of the corridor of $G'_2(k)$ (at the point $b \in B$) so that each edge (of $G'_1(k)$) at a crosses through a different edge (of $G'_2(k)$) near b. This ensures that a is not part of any $\mathfrak{c}$-cell. By similarly threading the vertices of $G'_2(k)$ through the corridors of $G'_3(k)$ and the vertices of $G'_3(k)$ through the corridors of $G'_1(k)$, $\mathfrak{c}$-freeness is ensured. The resulting drawing, as illustrated in Figure 9, is simple, and a quick calculation ensures the desired density. ◀

8 Concluding remarks

With this paper, we initiated the study of drawings of graphs with a forbidden type of cell. For arbitrary, non-homotopic, and simple drawings, and all cell types $\mathfrak{c}$ but $\square$, we determined whether graphs that admit a $\mathfrak{c}$-free drawing have at most linear or quadratic edge density. In many cases we obtain near-tight bounds. Being unable to control the $\square$-cell with our methods and constructions, we wonder in which regime the number of edges in $\square$-free drawings lies.

▶ **Open Problem 1.** *Is the edge density of simple graphs that admit a (simple, non-homotopic) $\square$-free drawing quadratic in n?*

Planar and 1-planar graphs can be characterized by forbidding not only one, but infinitely many types of cells. Perhaps there are other, less obvious, instances of beyond-planar graph classes characterized by forbidden cells. For example, in light of the density results on $\mathfrak{c}$-free and quasiplanar drawings (Section 6), the underlying graph classes might even coincide for non-homotopic drawings.

▶ **Open Problem 2.** *Does every graph that admits a non-homotopic*  *-free drawing also admit a non-homotopic quasiplanar drawing?*

The vast number of questions asked about other beyond planar graph classes – e.g. about complexity of recognition and stretchability – can be carried over to our setting. However, our forbidden patterns rely on the topology of concrete drawings rather than only abstract combinatorics of the graph or the drawing (such as crossing edge pairs). Hence, the resulting graph classes are not always closed under taking subgraphs. For example, K_5 admits a simple -free drawing while K_3 does not. In light of this, the results that almost all cell types can be avoided in drawings of K_n from Section 3 raise the question of which cell types can be avoided in drawings of any graph.

▶ **Open Problem 3.** *For which cell types* $\mathtt{c}$ *does every graph admit a (simple, non-homotopic)* $\mathtt{c}$*-free drawing?*

References

1 Eyal Ackerman. On the Maximum Number of Edges in Topological Graphs with no Four Pairwise Crossing Edges. *Discrete & Computational Geometry*, 41(3):365–375, 2009. doi:10.1007/s00454-009-9143-9.

2 Eyal Ackerman. On topological graphs with at most four crossings per edge. *Computational Geometry*, 85:101574, 2019. doi:10.1016/j.comgeo.2019.101574.

3 Eyal Ackerman and Gábor Tardos. On the maximum number of edges in quasi-planar graphs. *Journal of Combinatorial Theory Series A*, 114(3):563–571, 2007. doi:10.1016/J.JCTA.2006.08.002.

4 Patrizio Angelini, Michael A. Bekos, Henry Förster, and Michael Kaufmann. On RAC drawings of graphs with one bend per edge. *Theoretical Computer Science*, 828–829:42–54, 2020. doi:10.1016/j.tcs.2020.04.018.

5 Karin Arikushi, Radoslav Fulek, Balázs Keszegh, Filip Morić, and Csaba D. Tóth. Graphs that admit right angle crossing drawings. *Computational Geometry*, 45(4):169–177, 2012. doi:10.1016/j.comgeo.2011.11.008.

6 Jérémy Blanc. The best polynomial bounds for the number of triangles in a simple arrangement of n pseudo-lines. *Geombinatorics*, 21(1):5–14, 2011.

7 Otfried Cheong, Sariel Har-Peled, Heuna Kim, and Hyo-Sil Kim. On the Number of Edges of Fan-Crossing Free Graphs. *Algorithmica*, 73:673–695, 2015. doi:10.1007/s00453-014-9935-z.

8 Walter Didimo, Peter Eades, and Giuseppe Liotta. Drawing graphs with right angle crossings. *Theoretical Computer Science*, 412(39):5156–5166, 2011. doi:10.1016/j.tcs.2011.05.025.

9 Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani. A survey on graph drawing beyond planarity. *ACM Computing Surveys*, 52(1), 2019. doi:10.1145/3301281.

10 Stefan Felsner and Jacob E Goodman. Pseudoline arrangements. In *Handbook of Discrete and Computational Geometry*, pages 125–157. Chapman and Hall/CRC, 2017.

11 Stefan Felsner and Klaus Kriegel. Triangles in Euclidean Arrangements. *Discrete & Computational Geometry*, 22(3):429–438, 1999. doi:10.1007/PL00009471.

12 Branko Grünbaum. *Arrangements and Spreads*. Number 10 in Regional Conference Series in Mathematics. American Mathematical Society, 1972.

13 Heiko Harborth. Two-colorings of simple arrangements. In *Finite and Infinite Sets*, pages 371–378. North-Holland, 1984. doi:10.1016/B978-0-444-86893-0.50029-2.

14 Heiko Harborth. Some Simple Arrangements of Pseudolines with a Maximum Number of Triangles. *Annals of the New York Academy of Sciences*, 440(1):31–33, 1985. doi:10.1111/j.1749-6632.1985.tb14536.x.

15 Heiko Harborth and Ingrid Mengersen. Edges without crossings in drawings of complete graphs. *Journal of Combinatorial Theory, Series B*, 17(3):299–311, 1974. doi:10.1016/0095-8956(74)90035-5.

16 Michael Kaufmann, Boris Klemz, Kristin Knorr, Meghana M. Reddy, Felix Schröder, and Torsten Ueckerdt. The Density Formula: One Lemma to Bound Them All. In *32nd International Symposium on Graph Drawing and Network Visualization (GD 2024)*, volume 320 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.GD.2024.7.

17 Jesus Leanos, Mario Lomeli, Criel Merino, Gelasio Salazar, and Jorge Urrutia. Simple Euclidean Arrangements with No (≥ 5)-Gons. *Discrete & Computational Geometry*, 38(3):595–603, 2007.

18 Friedrich Levi. Die Teilung der projektiven Ebene durch Gerade oder Pseudogerade. *Ber. Math.-Phys. Kl. Sächs. Akad. Wiss*, 78:256–267, 1926.

19 János Pach, Rom Pinchasi, Micha Sharir, and Géza Tóth. Topological Graphs with No Large Grids. *Graphs and Combinatorics*, 21(3):355–364, 2005. doi:10.1007/s00373-005-0616-1.

20 Janos Pach, Rados Radoicic, Gabor Tardos, and Geza Toth. Improving the Crossing Lemma by Finding More Crossings in Sparse Graphs. *Discrete & Computational Geometry*, 36(4):527–552, 2006. doi:10.1007/s00454-006-1264-9.

21 János Pach and Géza Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17(3):427–439, 1997. doi:10.1007/BF01215922.

22 Jean-Pierre Roudneff. Quadrilaterals and pentagons in arrangements of lines. *Geometriae Dedicata*, 23(2), 1987. doi:10.1007/BF00181277.

Layered Polyline Drawings of Planar Graphs

Debajyoti Mondal
Department of Computer Science, University of Saskatchewan, Saskatoon, Canada

Abstract

A k-layer polyline drawing of a planar graph G is a planar drawing of G on a set L of k parallel lines such that each vertex is mapped to a point on L and each edge is mapped to a polygonal chain with the endpoints and bends lying on L. In the fixed embedding setting, the output drawing maintains the given planar embedding, whereas in the variable embedding setting, the embedding may change. Every n-vertex planar graph admits a polyline drawing on $2n/3$ layers, which is the best known upper bound for both settings. We improve this bound in the variable embedding setting. We show that every planar graph can be drawn on $14n/27 + O(\sqrt{n})$ layers by choosing a proper planar embedding, breaking the long-standing $2n/3$-layer barrier.

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics

Keywords and phrases Layered Drawing, Variable Embedding, Polyline Drawing, Cycle Separator

Digital Object Identifier 10.4230/LIPIcs.GD.2025.34

Acknowledgements The research is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

1 Introduction

A *polyline drawing* of a planar graph G maps each vertex of G to a distinct point in the Euclidean plane, and each edge of G to a polygonal chain such that no two edges cross except possibly at their common endpoints. A *straight-line drawing* is a special case of polyline drawing where every polygonal chain is a straight line segment. The *area* of a drawing Γ is the area of the smallest axis-aligned rectangle R that encloses Γ. The *width* and *height* of Γ are the width and height of R, respectively. Since a drawing can be scaled down arbitrarily, to measure the area we often require Γ to be a *grid drawing*, i.e., a drawing where all the vertices and bends have integer coordinates.

Drawing planar graphs on a small integer grid is an active research area in graph drawing [5, 6, 11, 13, 16, 27, 22, 25]. In this paper we focus on layered polyline drawings that attempt to reduce the height of the drawing while the other dimension could be unbounded. This can be seen as a drawing of a graph on a set of parallel lines or layers. Formally, a *k-layer polyline drawing* of a planar graph G is a planar drawing of G on a set L of k parallel lines (i.e., *layers*) such that each vertex is mapped to a point on L and each edge is mapped to a polygonal chain with the endpoints and bends lying on L. This drawing model has been studied extensively in the literature [4, 14, 17, 24, 26]. The distances between consecutive layers in a layered drawing can be assumed to be one unit, as any drawing with non-uniform spacing can be modified to satisfy this property without increasing the number of layers [17] (e.g., by adding extra bend points on edges at their intersections with layers and adjusting the gaps between layers). The results on the grid drawing of planar graphs readily imply upper bounds on the number of layers for their layered drawings. We briefly review these results under two drawing models. One is the *fixed embedding setting*, where the output drawing must respect the input planar embedding (i.e., a fixed combinatorial embedding with a prescribed outer face), and the other is the *variable embedding setting*, where the embedding in the output drawing can be different from the input embedding.


© Debajyoti Mondal;
licensed under Creative Commons License CC-BY 4.0

Fixed Embedding Setting. Every n-vertex planar graph admits a straight-line grid drawing with $O(n^2)$ area [11, 25]. Many algorithms have appeared in the literature that compute straight-line drawings [8, 9] and polyline drawings [6, 29] of planar graphs, where one dimension of the drawing (equivalently, number of layers) is bounded by $2n/3$. To provide an upper bound on area, drawing algorithms are typically presented on *maximal planar graphs* or *planar triangulations*, where each face of the given embedding is a cycle of three vertices. Similarly, throughout the paper, we will consider the input to be a planar triangulation. This upper bound on the number of layers is tight, that is, every polyline drawing of an n-vertex nested triangles graph requires at least $\lfloor 2(n-1)/3 \rfloor$ layers [6].

Variable Embedding Setting. The upper bound of $2n/3$ layers implied by the results in the fixed embedding setting [8, 6, 28] applies also to the variable embedding setting. To the best of our knowledge, it is not known whether a better bound can be achieved for arbitrary planar graphs in the variable embedding setting. However, better upper bounds are known in the variable embedding setting for several subclasses of planar graphs (e.g., outerplanar graphs [3], series-parallel graphs [31, 30], planar graphs with small maximum degree or small cycle separator [17], planar 3-trees [21, 20], and nested triangles graphs [18]). The best known lower bound on layers in the variable embedding setting is $\lfloor n/3 - 1 \rfloor$ [18]. The large gap between the upper and lower bounds motivates this paper.

Our Contribution. We develop an algorithm that can draw arbitrary n-vertex planar graphs on $14n/27 + O(\sqrt{n})$ layers in the variable embedding setting. The idea is to use a balanced cycle separator of size $O(\sqrt{n})$ to decompose the input planar graph into two subgraphs. We draw these subgraphs separately such that they can be merged without adding too many additional layers. The main challenge appears to be computing careful positions of the separator vertices in both drawings such that during the merge step, the vertices can be aligned with small modification. We tackle this by designing a drawing method that allows flexible positioning of separator vertices with necessary visibilities.

Note that the technique of using a separator to decompose a graph into subgraphs and then to merge the drawing of the subgraphs to obtain a drawing has previously been used in the literature [31, 15]. In particular, Durocher and Mondal [15] used a cycle separator of size $\lambda \in O(\sqrt{n})$ to draw an n-vertex planar graph in $4n/9 + O(\lambda\Delta)$ layers, where Δ is the maximum degree in the graph. If $\Delta \in o(\sqrt{n})$, the bound becomes $4n/9 + o(n)$ layers, but for $\Delta \geq \sqrt{n}$, it can become larger than n layers. When drawing each subgraph in $4n/9$ layers, their strategy was to contract the rest of the graph into a single vertex. However, the merge step required rerouting the edges incident to the separator vertices, which added an additional $O(\lambda\Delta)$ layers. Later, they used an edge separator of size $\lambda' \in O(\sqrt{n\Delta})$ instead of a cycle separator to improve the bound to $4n/9 + O(\lambda')$ [17]. However, both these additive terms $O(\lambda\Delta)$ and $O(\lambda')$ can sometimes be large (e.g., consider a wheel graph), and the total layer count may exceed n.

2 Preliminaries

A planar graph G is called *maximal* if the addition of any more edges to G results in a non-planar graph. Let Γ be a straight-line drawing of a maximal planar graph in $\mathbb{R}^2$. Then every face in Γ corresponds to a triangle in $\mathbb{R}^2$, and hence the graph is also called a *planar triangulation*. A planar graph is called *internally triangulated* if every inner face is a cycle of length three. A simple cycle C of G is called a *cycle separator* if both the interior and exterior

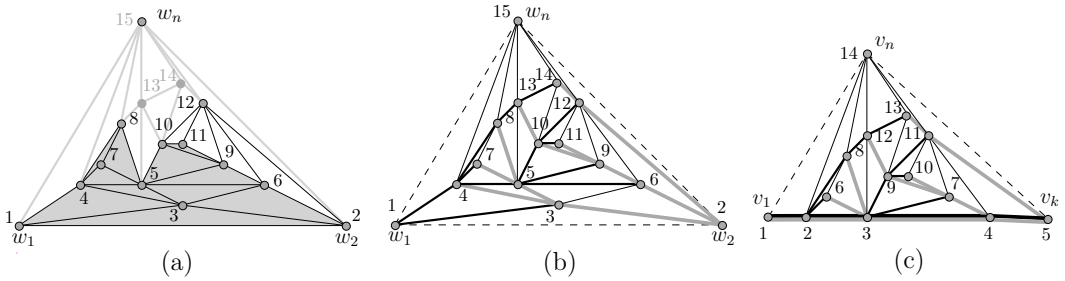


Figure 1 (a) Illustration for a canonical ordering. The graph G_{11} lies in the shaded region. (b) Corresponding Schnyder realizer. (c) Illustration for a Schnyder realizer of H.

of C contain at most $2n/3$ vertices. Every planar graph admits a simple cycle separator of size $O(\sqrt{n})$ and several studies attempt to improve the constant factor [12, 19, 23]. Throughout the paper we assume that the layers are horizontal and denote the x- and y-coordinates of a vertex v by $x(v)$ and $y(v)$, respectively.

Canonical Ordering and Schnyder Realizer. Let G be a planar triangulation, and let w_1, w_2 and w_n be the outer vertices of G in counter-clockwise order. Let $\sigma = (w_1, w_2, ..., w_n)$ be an ordering of all vertices of G. We denote by G_k, $3 \leq k \leq n$, the subgraph of G induced by $\{w_1, w_2, \ldots, w_k\}$, and by P_k, the clockwise path on the outer face of G_k that starts at w_1 and ends at w_2. The ordering σ is called a *canonical ordering* of G with respect to the outer edge (w_1, w_2) if for each k, $3 \leq k \leq n$, the following two conditions are satisfied [11]:

(a) G_k is 2-connected and internally triangulated.

(b) If $k+1 \leq n$, then w_{k+1} is an outer vertex of G_{k+1} and the neighbors of w_{k+1} in G_k appear consecutively on P_k.

For a vertex w_{k+1}, let P_k be the outer path $w_1, \ldots, w_l, \ldots, w_r, \ldots, w_2$ of G_k, where w_l and w_r are the leftmost and rightmost neighbors of w_{k+1}, respectively. We call the edges (w_l, w_{k+1}) and (w_{k+1}, w_r) the *l-edge* and the *r-edge* of w_{k+1}, respectively. The other edges incident to w_{k+1} in G_k are called the *m-edges* of w_{k+1}. For example, in Figure 1(a), the edges $(w_{12}, w_{10}), (w_{12}, w_2)$ are respectively the l-edge and r-edge of w_{12}. The m-edges of w_{12} are $(w_{12}, w_{11}), (w_{12}, w_9)$ and (w_{12}, w_6). The set of all m-edges in G corresponds to a tree T_m, which is rooted at w_n and spans all the inner vertices. Figure 1(b) illustrates T_m in thin solid lines. Similarly, the set of l-edges in G except (w_1, w_n) and (w_1, w_2) corresponds to a tree rooted at w_1, and the set of r-edges except (w_2, w_n) and (w_1, w_n) is a tree rooted at w_2. Figure 1(b) depicts the trees T_l and T_r in thick black and thick gray edges, respectively. The trees $\{T_l, T_r, T_m\}$ form the *Schnyder realizer* [25] of G. Each of T_l, T_r and T_m corresponds to a canonical ordering of G. Let $\text{leaf}(T_l), \text{leaf}(T_r)$ and $\text{leaf}(T_m)$ be the number of leaves in T_l, T_r and T_m, respectively.

▶ **Lemma 1** (Bonichon et al. [7]). *Let $\{T_l, T_r, T_m\}$ be a minimum Schnyder realizer of an n-vertex triangulation. Then $\text{leaf}(T_l) + \text{leaf}(T_r) + \text{leaf}(T_m) = 2n - \Delta$, where Δ is the number of cyclic inner faces in the realizer.*

Generalization to Internally Triangulated Graphs. The notions of canonical ordering and Schnyder realizer have been extended to other classes of planar graphs that are not necessarily triangulated [1, 2]. However, to describe our algorithm in this paper, we only need to consider the 2-connected and internally triangulated graphs. Let H be a 2-connected and internally

triangulated graph with the outer vertices $v_1, \ldots, v_k, v_n$ in counter-clockwise order. If $k > 2$, then consider a dummy edge (v_1, v_k) and triangulate the face $v_1, \ldots, v_k, v_1$. We can now order the inner vertices of H satisfying properties (a) and (b) of a canonical ordering. We can define the l- and r-edges in the same way as we defined for canonical ordering. Observe that the set of m-edges corresponds to a tree T_m, which is rooted at v_n, but the l-edges and r-edges may no longer form the trees T_l and T_r. To form the trees T_l and T_r, we assume the edges on the outer path $v_1, v_2, \ldots, v_k$ to belong to both these trees, e.g., Figure 1(c). Let Γ be a drawing of H. We call a vertex v of H to have *top visibility* in Γ if the vertically upward ray starting at v does not intersect Γ except at v.

We now have the following lemma, which will be used later to describe our drawing algorithm.

▶ **Lemma 2.** *Let H be a 2-connected and internally triangulated graph with the outer vertices $v_1, \ldots, v_k, v_n$ in counter-clockwise order. Let T_l, T_r, T_m be a Schnyder realizer of H rooted at v_1, v_k, v_n, respectively, where $v_1, \ldots, v_k$ is a path in both T_l and T_r. Then H admits the following types of layered drawings:*

(a) *A drawing on* $\text{leaf}(T_l)$ *layers where the path $v_1, \ldots, v_k$ is drawn on the bottommost layer and v_1, v_k have top visibility.*

(b) *A drawing on* $\text{leaf}(T_r)$ *layers where the path $v_1, \ldots, v_k$ is drawn on the bottommost layer and v_1, v_k have top visibility.*

(c) *A drawing on* $\text{leaf}(T_m) + k$ *layers where the path $v_1, \ldots, v_k$ lies along a vertical line L (perpendicular to the layers) and all other vertices and edges lie in the same half-plane of L.*

Proof. Here we show that H admits a drawing on a $(\text{leaf}(T_m) + k) \times \text{leaf}(T_l)$ grid where the path $v_1, \ldots, v_k$ is drawn on the bottommost layer and v_1, v_k has top visibility. This will correspond to the type-(a) drawing of the lemma, whereas the type-(c) drawing can be obtained by rotating the drawing 90°. To obtain the drawing of type-(b), one can construct drawing on a $(\text{leaf}(T_m) + k) \times \text{leaf}(T_r)$ grid symmetrically.

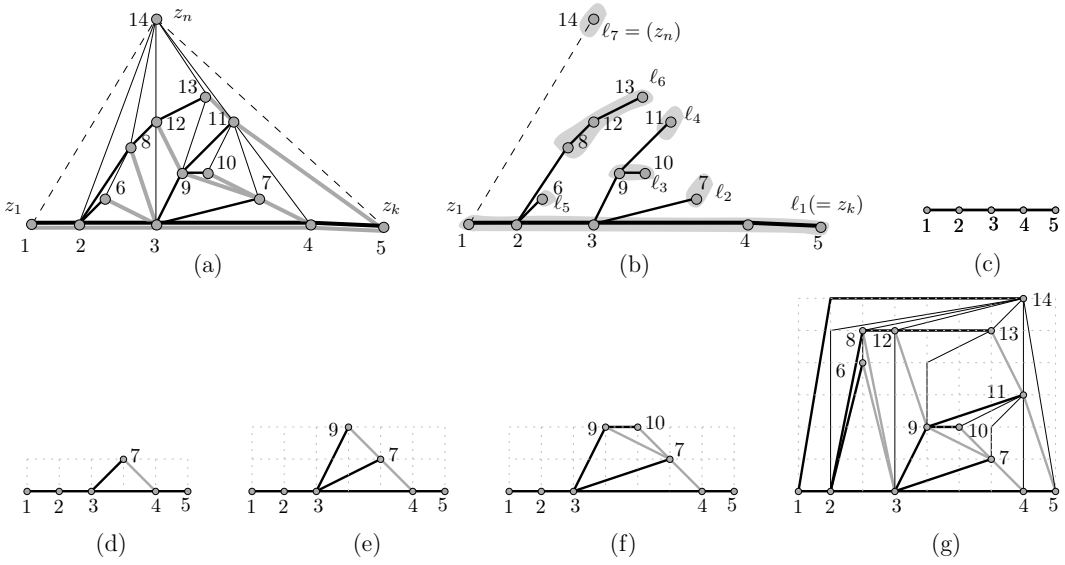


Figure 2 (a) A planar graph H. (b) Pseudo-segments of T_l are shaded in gray. (c)–(f) Illustration for some vertex insertions in the order they appear in σ. (g) The final polyline drawing of H.

We assume familiarity with the "shift algorithm" to compute grid drawings for planar graphs [11]. We treat the vertex v_n as a leaf of T_l. Let $z_1(= v_1), \ldots, z_k(= v_k)$, $z_{k+1}, \ldots, z_n(= v_n)$ be the vertices listed according to a preorder traversal of T_l, where the children are visited in counter-clockwise order. Then the ordering $\sigma = (z_{k+1}, \ldots, z_n)$ is another canonical ordering of G [10, Lemma 3.5]. Let $\ell_1(= z_k), \ell_2, \ldots, \ell_s(= z_n)$ be the leaves of T_l according to the preorder traversal of T_l, where s is the number of leaves in T_l. We assign each leaf of T_l a path called *pseudo-segment*, as follows. The first *pseudo-segment* $\mathcal{S}_1$ *assigned to* ℓ_1 is the unique path from z_1 to $l_1(= z_k)$. The *pseudo-segment* $\mathcal{S}_j$ *assigned to* ℓ_j, where $1 < j \leq s$, is the shortest path in T_l that starts at ℓ_j and ends at a vertex which is adjacent to some previous pseudo-segment. Figure 2(a)–(b) illustrates such a pseudo-segment decomposition of T_l. Let H_q be the subgraph of H induced by the vertices $z_1, \ldots, z_q$, where $1 \leq q \leq n$. We now show that H_q admits a drawing Γ_q satisfying the following properties.

- I_1. Γ_q is a polyline drawing on $(\delta + k) \times j$ layers, where δ is the number of leaves of T_m in H_q and z_q belongs to the pseudo-segment $\mathcal{S}_j$.
- I_2. The y-coordinates of the vertices of a pseudo-segment $\mathcal{S}_i$ is i, where $1 \leq i \leq j$.
- I_3. The clockwise path P_q on the outer face of H_q is drawn as a strictly x-monotone polygonal chain in Γ_q, i.e., the vertices on P_q have top visibility.

We first draw the path $z_1(= v_1), \ldots, z_k(= v_k)$ by placing each z_i, where $1 \leq i \leq k$, at $(i, 1)$ (Figure 2(c)). It is straightforward to verify I_1–I_3 for Γ_k. We now add the remaining vertices in the order they appear in σ. While inserting a new vertex z_t, where $t > k$, we consider two cases depending on whether z_t is a leaf or an internal vertex in T_m. Let $P_t = (z_1, \ldots, z_t^l, z_t, z_t^r, \ldots, z_k)$ be the clockwise path on the outer face of H_t. Let $\mathcal{S}_{j'}$ be the pseudo-segment that contains z_t.

Case 1. If z_t is a leaf of T_m, then we shift the vertices $z_t^r, \ldots, z_k$ and their descendants in T_m one unit to the right, e.g., see the vertex insertions in Figure 2(d)–(f). Then we place z_t at $(x(z_t^r) - 1, j')$. Note that z_t^l and z_t^r have top visibility in Γ_{t-1} and retain this visibility even after the shift. Since j' is the highest y-coordinate in the current drawing and since $x(z_t) = x(z_t^r) - 1$, we can draw the edge (z_t^r, z_t) with a straight line segment without introducing any edge crossing. If z_t^l belongs to $\mathcal{S}_{j'}$ or if $x(z_t) = x(z_t^l) + 1$, then we can draw (z_t, z_t^r) with a straight line segment. Otherwise, we use a bend point at $(x(z_t^l) + 1, j')$.

We now show that Γ_t satisfies properties I_1–I_3. Here I_1 holds as the drawing width increases by one unit due to the insertion of a new leaf and the height of the drawing becomes j'. The shift operation keeps the resulting drawing planar, does not change the y-coordinates of the vertices, and does not affect the x-monotonicity of the edges [11]. Therefore, the drawing of the l- and r-edges of z_t ensures that the path (z_t^l, z_t, z_t^r) is drawn as an x-monotone polygonal chain where the vertices z_t^l, z_t, z_t^r have top visibility. Consequently, the drawing Γ_t satisfies I_2 and I_3.

Case 2. If z_t is an internal vertex of T_m, then we place z_t at $(x(z_t^r) - 1, j')$. Let (z_t, z') be the first m-edge of z_t in clockwise order. Since P_{t-1} is strictly x-monotone in Γ_{t-1}, the other vertices on the pseudo-segment $\mathcal{S}_{j'}$ before z_t lie to the left of the line $x = x(z')$. Since z' is a child of v in T_m, it cannot belong to $\mathcal{S}_{j'}$ and hence its y-coordinate is smaller than j'. Hence the grid point $(x(z_t^r) - 1, j')$ is empty.

Since $x(z_t) = x(z_t^r) - 1$, we can draw (z_t, z_t^r) with a straight line segment (e.g., insertion of z_{14} in Figure 2(g)). If z_t^l belongs to $\mathcal{S}_{j'}$ or if $x(z_t) = x(z_t^l) + 1$, then we can draw (z_t, z_t^l) with a straight line segment. Otherwise, we use a bend point at $(x(z_t^l) + 1, j')$ (Figure 2(g)). We now consider the drawing of the m-edges. To draw an m-edge (z_t, w), we first check whether $y(w) = j' - 1$. If so, then we can draw the edge with a straight line segment. Otherwise, we

create a bend point at $(x(w), j'-1)$. This is possible because w appears in a pseudo-segment $\mathcal{S}_{j''}$ where $j'' < j$, and thus the grid point $(x(w), j'-1)$ is empty. Here I_1 holds as we do not change the width of the current drawing and the height increases only when we start adding a new pseudo-segment. I_2–I_4 hold by the drawing of the l- and r-edges of z_t ensuring the top visibility for z_t^l, z_t, z_t^r, and by the property of the shift algorithm that it preserves the y-coordinates of the vertices and the x-monotonicity of the edges [11]. ◀

3 Drawing Algorithm

We first choose a suitable embedding of G, then decompose the graph into two subgraphs and draw them independently, and finally, merge these drawings by considering different cases depending on the structure of their Schnyder realizers.

Embedding Selection. Let G be a planar triangulation with a simple cycle separator $C = (v_1, \ldots, v_k)$ of size $k = O(\sqrt{n})$, as shown in bold in Figure 3(a). Let Γ be a planar embedding of G and let (a, b, c) be an inner face in Γ such that $(a, b) = (v_1, v_k)$ is an edge of C and c is an inner vertex in Γ. We then create a new planar embedding Γ' by choosing (a, b, c) as the outer face (Figure 3(b)). We then add a subdivision vertex d on the edge (a, b) (Figure 3(c)), and triangulate the inner face incident to d. Finally, for each pair of vertices v_i, v_j in C, where $1 \leq i+1 < j \leq k$, if the edge (v_i, v_j) exists in G, then we add a subdivision vertex on (v_i, v_j) and triangulate the graph. Let G' be the resulting drawing. By G_i (G_o) we refer to the subgraph of G' that lies in the closed interior (exterior) of C. By n_o and n_i we refer to the number of vertices of G_o and G_i, respectively. Since we added at most $O(\sqrt{n})$ division vertices, $n_o, n_i \leq 2n/3 + O(\sqrt{n})$.

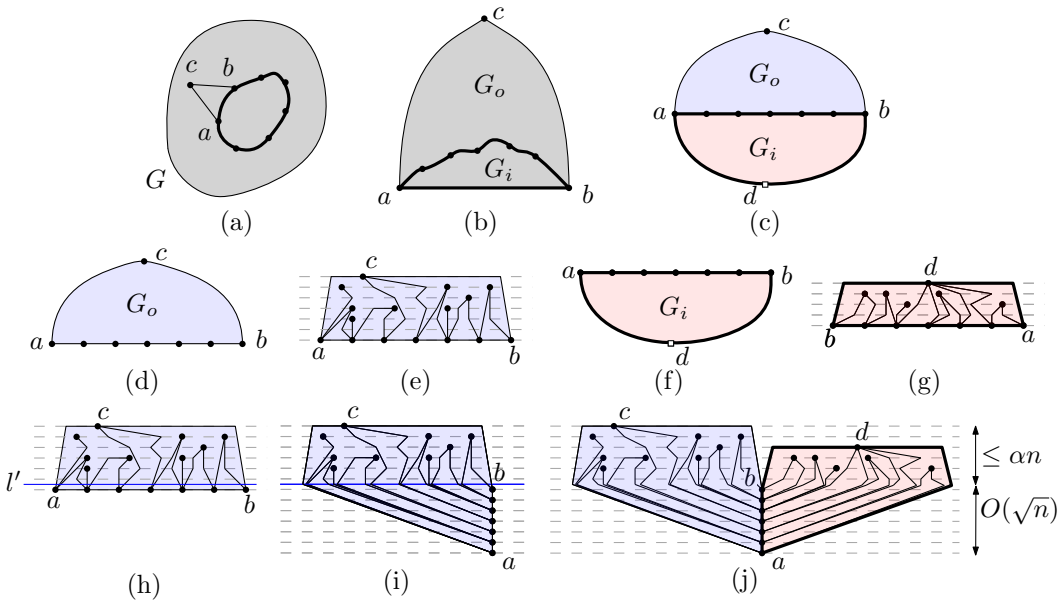


Figure 3 (a) A planar graph G with a cycle separator shown in bold. (b) Computation of the required embedding. (c) An illustration for G'. (d)–(e) A drawing of G_o, where only a few edges adjacent to the vertices of $a, \ldots, b$ are shown. (f)–(g) A drawing of G_i. (h)–(i) Modification of the drawing of G_o. (j) The final drawing of G' obtained by merging the drawings of G_o and G_i.

Draw and Merge. Let T_l^o, T_r^o, T_m^o be the trees of a minimum Schnyder realizer of G_o rooted at a, b, c, respectively. Similarly, let T_l^i, T_r^i, T_m^i be the trees of a minimum Schnyder realizer of G_i rooted at b, a, d, respectively. Let $\alpha \in (0, 2/3]$ be a positive constant to be determined later. We now consider two cases depending on whether the following condition holds: $\min\{\text{leaf}(T_l^o), \text{leaf}(T_r^o)\} \leq \alpha n$ and $\min\{\text{leaf}(T_l^i), \text{leaf}(T_r^i)\} \leq \alpha n$.

Case 1 ($\min\{\text{leaf}(T_l^o), \text{leaf}(T_r^o)\} \leq \alpha n$ and $\min\{\text{leaf}(T_l^i), \text{leaf}(T_r^i)\} \leq \alpha n$). In this case we construct a drawing of G' on $\alpha n + O(\sqrt{n})$ layers, as follows. By Lemma 2, G_o admits a drawing on $\min\{\text{leaf}(T_l^o), \text{leaf}(T_r^o)\} \leq \alpha n$ layers with the path $v_1(=a), v_2, \ldots, v_k(=b)$ of the cycle separator drawn on the bottommost layer (Figure 3(d)–(e)). Let l_0 be the bottommost layer and let l_1 be the layer above l_0. We consider a horizontal line l' between these two layers (i.e., the blue line in Figure 3(h)) and add a bend point at each intersection point between l' and the edges. We insert $k-1$ layers below l_0 and move the vertices $v_{k-1}, \ldots, v_2, v_1(=a)$ on these newly inserted layers consecutively below b, as shown in Figure 3(h)–(i). We redraw the edges and part of the edges below l' with straight line segments. Since the ordering of the bend points on l' corresponds to the ordering of the vertices $a, \ldots, b$ in the newly inserted layers, the resulting drawing remains planar. Since $k+1 = O(\sqrt{n})$, the drawing takes at most $\alpha n + O(\sqrt{n})$ layers. We draw G_i using Lemma 2 on $\min\{\text{leaf}(T_l^i), \text{leaf}(T_r^i)\} \leq \alpha n$ layers with the path $v_1(=a), v_2, \ldots, v_k(=b)$ of the cycle separator drawn on the bottommost layer (Figure 3(f)–(g)). We then modify the drawing to bring the vertices $v_{k-1}, \ldots, v_1(=a)$ on a set of $(k-1)$ consecutive bottommost layers of the drawing below b. To obtain the final drawing of G', we bring the drawings of G_o and G_i close together such that they coincide on the path $a, \ldots, b$ (Figure 3(j)). The top visibility of b ensures planarity, i.e., the polylines representing (c, b) and (d, b) do not overlap. Since these drawings share a common set of layers, the number of layers is bounded by $\alpha n + O(\sqrt{n})$.

Case 2 ($\min\{\text{leaf}(T_l^o), \text{leaf}(T_r^o)\} > \alpha n$ or $\min\{\text{leaf}(T_l^i), \text{leaf}(T_r^i)\} > \alpha n$). Without loss of generality assume that $\min\{\text{leaf}(T_l^o), \text{leaf}(T_r^o)\} > \alpha n$. Then by Lemma 1, we have $\text{leaf}(T_m^o) \leq 2n_o - 2\alpha n$. In this scenario, we consider two subcases depending on whether $\text{leaf}(T_m^i) \leq 2n_i/3$ or not, and in both cases we construct a drawing of G' on $2n_o - 2\alpha n + O(\sqrt{n}) + 2n_i/3$ layers.

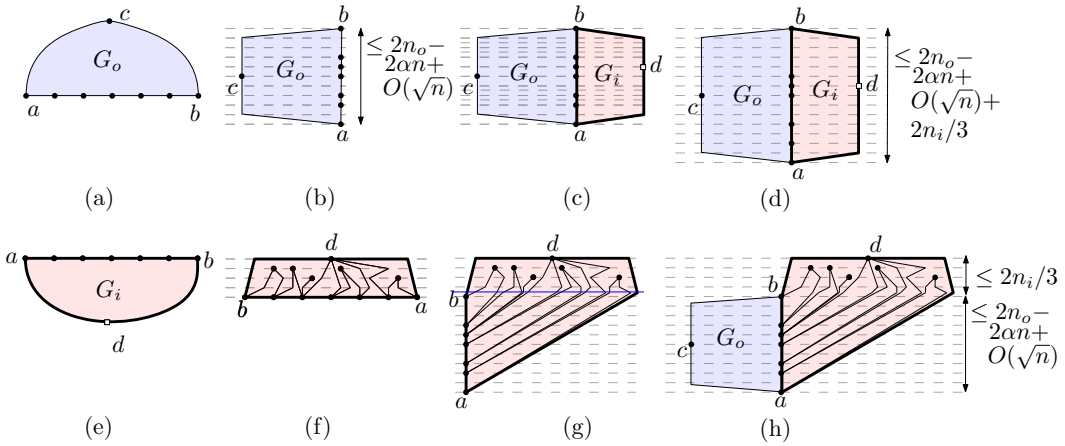


Figure 4 Illustration for (a)–(d) Case 2.1 and (e)–(h) Case 2.2.

Case 2.1 ($\mathbf{leaf}(T_m^i) \leq 2n_i/3$). We first draw G_o using Lemma 2 in $\text{leaf}(T_m^o) + k \leq 2n_o - 2\alpha n + O(\sqrt{n})$ layers where the path $v_1, \ldots, v_k$ is drawn along a vertical line L (perpendicular to the layers) and the drawing lies to the left half-plane of L (Figure 4(a)–(b)). We now draw G_i on the right half-plane of L taking the drawing of $v_1, \ldots, v_k$ as input, and inserting the remaining vertices of G_i following the method described in the proof of Lemma 2. This inserts new layers within the drawing of G_o. Since $\text{leaf}(T_m^i) \leq 2n_i/3$, the total number of layers remains bounded by $2n_o - 2\alpha n + O(\sqrt{n}) + 2n_i/3$ (Figure 4(c)–(d)).

Case 2.2 ($\mathbf{leaf}(T_m^i) > 2n_i/3$). We draw G_o in the same way as in Case 2.1 on $2n_o - 2\alpha n + O(\sqrt{n})$ layers. By Lemma 1, either $\text{leaf}(T_l^i)$ or $\text{leaf}(T_r^i)$ is at most $2n_i/3$. By Lemma 2, G_i admits a drawing on $2n_i/3$ layers with the path $v_1(= a), v_2, \ldots, v_k(= b)$ drawn on the bottommost layer l_0 (Figure 4(e)–(f)). We modify the drawing similar to Case 1 by inserting new layers below l_0 and moving the vertices $v_{k-1}, \ldots, v_1(= a)$ below b. However, instead of placing these vertices on consecutive layers, we move them to their corresponding layers in the drawing of G_o. Hence the total number of layers after merging the drawings of G_o and G_i is bounded by $2n_o - 2\alpha n + O(\sqrt{n}) + 2n_i/3$ (Figure 4(g)–(h)).

Upper Bound Computation. The algorithm produces a drawing on $\max\{\alpha n + O(\sqrt{n}), 2n_o - 2\alpha n + O(\sqrt{n}) + 2n_i/3\}$ layers. Since $n_o + n_i = n + O(\sqrt{n})$ and $n_o \leq 2n/3 + O(\sqrt{n})$, we have

$$\begin{aligned} 2n_o - 2\alpha n + O(\sqrt{n}) + 2n_i/3 &= 2(n_o + n_i)/3 + 4n_o/3 - 2\alpha n + O(\sqrt{n}) \\ &\leq 2n/3 + 8n/9 - 2\alpha n + O(\sqrt{n}) \\ &\leq 14n/9 - 2\alpha n + O(\sqrt{n}). \end{aligned}$$

Hence the number of layers is bounded by $\max\{\alpha n + O(\sqrt{n}), 14n/9 - 2\alpha n + O(\sqrt{n})\}$ layers, which is minimized when $\alpha = 14/27$. We thus have the following theorem.

▶ **Theorem 3.** *Every planar graph with n vertices admits a planar polyline drawing on $14n/27 + O(\sqrt{n})$ layers.*

4 Conclusion

In this paper we have shown how to draw an n-vertex planar graph using $14n/27 + O(\sqrt{n})$ layers. Since only a $\lfloor n/3 - 1 \rfloor$ lower bound is known [18], a natural open problem is to reduce this gap. It would be interesting to investigate whether our strategy can be combined with the techniques in [15, 17] to improve the upper bound further.

We did not analyze the number of bends produced in our drawing, which could be another parameter for optimization. The trade-off between the number of layers and the number of bends also requires further investigation.

References

1 Luca Castelli Aleardi, Éric Fusy, and Thomas Lewiner. Schnyder woods for higher genus triangulated surfaces, with applications to encoding. *Discrete & Computational Geometry*, 42(3):489–516, 2009. doi:10.1007/s00454-009-9169-z.

2 Olivier Bernardi and Éric Fusy. Schnyder decompositions for regular plane graphs and application to drawing. *Algorithmica*, 62(3-4):1159–1197, 2012. doi:10.1007/s00453-011-9514-5.

3 Therese C. Biedl. Small drawings of outerplanar graphs, series-parallel graphs, and other planar graphs. *Discrete & Computational Geometry*, 45(1):141–160, 2011. doi:10.1007/s00454-010-9310-z.

4 Therese C. Biedl. Transforming planar graph drawings while maintaining height. *CoRR*, abs/1308.6693, 2013. doi:10.48550/arXiv.1308.6693.

5 Therese C. Biedl. On area-optimal planar graph drawings. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 8572 of *Lecture Notes in Computer Science*, pages 198–210. Springer, 2014. doi:10.1007/978-3-662-43948-7_17.

6 Nicolas Bonichon, Bertrand Le Saëc, and Mohamed Mosbah. Optimal area algorithm for planar polyline drawings. In *Proceedings of the 28th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 2573 of *LNCS*, pages 35–46. Springer, 2002. doi:10.1007/3-540-36379-3_4.

7 Nicolas Bonichon, Bertrand Le Saëc, and Mohamed Mosbah. Wagner's theorem on realizers. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2380 of *Lecture Notes in Computer Science*, pages 1043–1053. Springer, 2002. doi:10.1007/3-540-45465-9_89.

8 Franz J. Brandenburg. Drawing planar graphs on $\frac{8}{9}n^2$ area. *Electronic Notes in Discrete Mathematics*, 31:37–40, 2008. doi:10.1016/j.endm.2008.06.005.

9 Marek Chrobak and Shin-Ichi Nakano. Minimum-width grid drawings of plane graphs. *Computational Geometry*, 11(1):29–54, 1998. doi:10.1016/S0925-7721(98)00016-9.

10 Hubert de Fraysseix and Patrice Ossona de Mendez. On topological aspects of orientations. *Discrete Mathematics*, 229(1-3):57–72, 2001. doi:10.1016/S0012-365X(00)00201-6.

11 Hubert de Fraysseix, János Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990. doi:10.1007/BF02122694.

12 Hristo N. Djidjev and Shankar M. Venkatesan. Reduced constants for simple cycle graph separation. *Acta Informatica*, 34(3):231–243, 1997. doi:10.1007/s002360050082.

13 Danny Dolev, Frank Thomson Leighton, and Howard Trickey. Planar embedding of planar graphs. *Advances in Computing Research*, 2:147–161, 1984. URL: https://hdl.handle.net/1721.1/149047.

14 Vida Dujmović, Michael R Fellows, Matthew Kitching, Giuseppe Liotta, Catherine McCartin, Naomi Nishimura, Prabhakar Ragde, Frances Rosamond, Sue Whitesides, and David R Wood. On the parameterized complexity of layered graph drawing. *Algorithmica*, 52:267–292, 2008. doi:10.1007/s00453-007-9151-1.

15 Stephane Durocher and Debajyoti Mondal. Drawing planar graphs with reduced height. In *International Symposium on Graph Drawing*, pages 392–403. Springer, 2014. doi:10.1007/978-3-662-45803-7_33.

16 Stephane Durocher and Debajyoti Mondal. Trade-offs in planar polyline drawings. In *Proceedings of the 22nd International Symposium on Graph Drawing (GD)*, volume 8871 of *LNCS*, pages 306–318. Springer, 2014. doi:10.1007/978-3-662-45803-7_26.

17 Stephane Durocher and Debajyoti Mondal. Drawing planar graphs with reduced height. *J. Graph Algorithms Appl.*, 21(4):433–453, 2017. doi:10.7155/jgaa.00424.

18 Fabrizio Frati and Maurizio Patrignani. A note on minimum area straight-line drawings of planar graphs. In *Proceedings of the 15th International Symposium on Graph Drawing (GD)*, volume 4875 of *LNCS*, pages 339–344. Springer, 2008. doi:10.1007/978-3-540-77537-9_33.

19 Hillel Gazit and Gary L Miller. Planar separators and the Euclidean norm. In *Proceedings of the International Symposium on Algorithms (SIGAL)*, volume 450 of *LNCS*, pages 338–347. Springer, 1990. doi:10.1007/3-540-52921-7_83.

20 Md. Iqbal Hossain, Debajyoti Mondal, Md. Saidur Rahman, and Sammi Abida Salma. Universal line-sets for drawing planar 3-trees. In *Proceedings of the 6th International Workshop on Algorithms and Computation (WALCOM)*, volume 7157 of *LNCS*, pages 136–147. Springer, 2012. doi:10.1007/978-3-642-28076-4_15.

21 Md Iqbal Hossain, Debajyoti Mondal, Md Saidur Rahman, and Sammi Abida Salma. Universal line-sets for drawing planar 3-trees. *Journal of Graph Algorithms and Applications*, 17(2):59–79, 2013. doi:10.7155/jgaa.00285.

22 Charles E. Leiserson. Area-efficient graph layouts (for VLSI). In *Proceedings of the 21st Annual Symposium on Foundations of Computer Science*, pages 270–281. IEEE Computer Society, 1980. doi:10.1109/SFCS.1980.13.

23 Gary L. Miller. Finding small simple cycle separators for 2-connected planar graphs. *J. Comput. Syst. Sci.*, 32(3):265–279, 1986. doi:10.1016/0022-0000(86)90030-9.

24 Debajyoti Mondal, Muhammad Jawaherul Alam, and Md Saidur Rahman. Minimum-layer drawings of trees. In *Proceedings of the 5th International Workshop on Algorithms and Computation*, volume 6552 of *LNCS*, pages 221–232. Springer, 2011. doi:10.1007/978-3-642-19094-0_23.

25 Walter Schnyder. Embedding planar graphs on the grid. In *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), San Francisco, California, USA*, pages 138–148. ACM, 1990. URL: http://dl.acm.org/citation.cfm?id=320176.320191.

26 Matthew Suderman. Pathwidth and layered drawings of trees. *International Journal of Computational Geometry and Applications*, 14:203–225, 2004. doi:10.1142/S0218195904001433.

27 Roberto Tamassia, editor. *Handbook of Graph Drawing and Visualization*. CRC Press, 2013.

28 Huaming Zhang. Planar polyline drawings via graph transformations. *Algorithmica*, 57(2):381–397, 2010. doi:10.1007/s00453-008-9215-x.

29 Huaming Zhang and Sadish Sadasivam. On planar polyline drawings. In *Proceedings of the 15th International Symposium on Graph Drawing (GD)*, pages 213–218. Springer, 2007. doi:10.1007/978-3-540-77537-9_22.

30 Xiao Zhou, Takashi Hikino, and Takao Nishizeki. Small grid drawings of planar graphs with balanced bipartition. In *Proceedings of the 4th International Workshop on Algorithms and Computation (WALCOM)*, volume 5942 of *LNCS*, pages 47–57. Springer, 2010. doi:10.1007/978-3-642-11440-3_5.

31 Xiao Zhou, Takashi Hikino, and Takao Nishizeki. Small grid drawings of planar graphs with balanced partition. *Journal of Combinatorial Optimization*, 24(2):99–115, 2012. doi:10.1007/s10878-011-9381-7.

A Walk on the Wild Side: A Shape-First Methodology for Orthogonal Drawings

Giordano Andreola
Roma Tre University, Italy

Susanna Caroppo
Roma Tre University, Italy

Giuseppe Di Battista
Roma Tre University, Italy

Fabrizio Grosso
CeDiPa - University of Perugia, Italy

Maurizio Patrignani
Roma Tre University, Italy

Allegra Strippoli
Roma Tre University, Italy

Abstract

Several algorithms for the construction of orthogonal drawings of graphs, including those based on the Topology-Shape-Metrics (TSM) paradigm, tend to prioritize the minimization of crossings. This emphasis has two notable side effects: some edges are drawn with unnecessarily long sequences of segments and bends, and the overall drawing area may become excessively large. As a result, the produced drawings often lack geometric uniformity. Moreover, orthogonal crossings are known to have a limited impact on readability, suggesting that crossing minimization may not always be the optimal goal. In this paper, we introduce a methodology that "subverts" the traditional TSM pipeline by focusing on minimizing bends. Given a graph G, we ideally seek to construct a rectilinear drawing of G, that is, an orthogonal drawing with no bends. When not possible, we incrementally subdivide the edges of G by introducing dummy vertices that will (possibly) correspond to bends in the final drawing. This process continues until a rectilinear drawing of a subdivision of the graph is found, after which the final coordinates are computed. We tackle the (NP-complete) rectilinear drawability problem by encoding it as a SAT formula and solving it with state-of-the-art SAT solvers. If the SAT formula is unsatisfiable, we use the solver's proof to determine which edge to subdivide. Our implementation, DOMUS, which is fairly simple, is evaluated through extensive experiments on small- to medium-sized graphs. The results show that it consistently outperforms OGDF 's TSM-based approach across most standard graph drawing metrics.

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

Keywords and phrases Non-planar Orthogonal Drawings, SAT Solver, Experimental Comparison

Digital Object Identifier 10.4230/LIPIcs.GD.2025.35

Related Version *Full Version*: `https://arxiv.org/abs/2508.19416` [1]

Funding This research was supported, in part, by MUR of Italy (PRIN Project no. 2022ME9Z78 – NextGRAAL and PRIN Project no. 2022TS4Y3N – EXPAND). The fourth author was supported by Ce.Di.Pa. - PNC Programma unitario di interventi per le aree del terremoto del 2009-2016 - Linea di intervento 1 sub-misura B4 - "Centri di ricerca per l'innovazione" CUP J37G22000140001.



© Giordano Andreola, Susanna Caroppo, Giuseppe Di Battista, Fabrizio Grosso, Maurizio Patrignani, and Allegra Strippoli;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

Orthogonal drawings, where edges are represented as chains of horizontal and vertical segments, are a foundational topic in Graph Drawing, valued both for their practical applications and for the rich body of research they have inspired over the past four decades.

The most well-known and widely studied methodology for constructing orthogonal drawings is the *Topology-Shape-Metrics* (*TSM*) approach (see, e.g., [19, 42, 43]). TSM generates an orthogonal drawing in three steps. First, a planar embedding (*topology*) of the graph is computed, and *dummy vertices* are possibly introduced to represent edge crossings. Second, a *shape* – minimizing the number of *bends* (points where two edge segments meet at a right angle) – is computed, see [40]. Third, an orthogonal drawing (*metrics*) consistent with the computed shape is constructed. Although originally designed for graphs with maximum degree 4, TSM has been extended to general graphs by either representing vertices as boxes sized proportionally to their degree (see, e.g., [4, 5]), or by allowing partial edge overlaps, which requires adapting the flow-based optimization accordingly (see, e.g., [22, 24]).

While TSM is arguably the most well-known method, alternative approaches have been proposed for orthogonal drawings (see, e.g., [10, 11, 33, 34, 39]). Nonetheless, the comprehensive and still influential quantitative aggregate [3] evaluation in [20] shows that the TSM approach outperforms all the above cited alternative methods across nearly all standard graph drawing metrics. Note that other algorithms (see e.g., [28]) have been proposed and assessed not through quantitative measures but via user studies. Also, the algorithm in [26] neglects the experiments in [20] in favor of a comparison with [28].

Furthermore, the success of TSM is underscored by its implementation in widely used open-source libraries such as OGDF [13] and GDToolkit [18], as well as its adoption in the Graph Drawing engine of yWorks [22, 24, 29, 36], a leading company in graph visualization.

However, two key observations can be made. (1) Existing algorithms, including those based on TSM, prioritize minimizing crossings. This often leads to unnecessarily long edge paths with many bends and results in drawings with large area and poor geometric uniformity. (2) Several studies (e.g., [27]) have shown that orthogonal crossings, where edges meet at right angles, have minimal impact on readability. This suggests that aggressively minimizing crossings in orthogonal drawings may not always be the optimal design goal.

In this paper, we start from the two aforementioned observations and introduce a methodology, called *Shape-Metrics* (*SM*), that "subverts" the traditional TSM pipeline. SM focuses first on bends. Namely, given a graph G, we ideally seek to construct a *rectilinear drawing* of G, that is, a drawing in which all edges are represented by straight horizontal or vertical segments with no bends. When such a drawing is not feasible, we incrementally subdivide the edges of G by introducing dummy vertices. Each of these vertices will (possibly) correspond to a bend in the final drawing. We continue this process until we obtain a subdivision of G that admits a rectilinear drawing. Once such a subdivision is found, the final metric phase computes the actual coordinates of the drawing.

SM is based on the concept of *shaped* graph, which we define as an assignment of a label from the set $\mathcal{L} = \{L, R, D, U\}$ to each of its edges, indicating the direction – Left, Right, Down, or Up – of that edge in a drawing. Given a shaped graph, it can be decided in polynomial time [30] if it admits a rectilinear drawing with that shape. On the other hand, checking if a graph admits a rectilinear drawable shape is NP-complete [21]. We exploit a new simple characterization of rectilinear drawable shaped graphs. Informally, a shaped cycle is said to be *complete* if its edges have all the four labels; we show that a shaped graph is rectilinear drawable if and only if all its cycles are complete.

We tackle the NP-hardness barrier by formulating the rectilinear drawability testing problem as a Boolean satisfiability (SAT) problem and leveraging the power of modern SAT solvers. SAT solvers have already proven to be effective in combinatorics and geometry (see, e.g., [12, 17, 31, 37, 38]). However, as far as we know, SAT solvers have rarely been applied to Graph Drawing. The related works we are aware of are [9], which employs a *hybrid* ILP/SAT approach to compute *st*-orientations and visibility representations, [8] where a SAT solver is used to compute book embeddings, and [14, 15, 16] where a SAT solver is used to test upward planarity.

There are differences and analogies between SM and TSM. TSM leverages the fact that checking if a graph is planar can be done efficiently. On our side we have that, unfortunately, checking if a graph is rectilinear drawable is NP-complete [21]. On the other hand both methodologies use specific properties of shapes: TSM's cornerstone is the balance of turns in shapes of faces of planar graphs [44], while SM's cornerstone is the completeness of cycles. Also, both methodologies augment the graph to be drawn with dummy vertices, TSM to represent crossings, SM to represent bends.

We implemented SM in a tool called DOMUS (*Drawing Orthogonal Metrics Using the Shape*) and experimented its effectiveness against the TSM implementation available in OGDF [13] measuring widely adopted Graph Drawing metrics [3, 20]. Although other comparable tools exist (e.g., GDToolkit [18]), we selected OGDF as our benchmark due to its open-source nature, widespread adoption within the Graph Drawing community, and the fact that it has benefited from over a decade of active development and optimization. In view of the results in [20] it was pointless to compare DOMUS with the algorithms in [10, 11, 33, 34, 39]. Also, the algorithms in [26, 28] adopt a definition of orthogonal drawing which is different from the widely adopted one given in this paper. E.g., even if a vertex has degree less or equal than 4 more than one of its incident edges can exit from the same direction. Hence, comparing DOMUS with them would be unfair.

We did two sets of experiments, which we refer to as "in-vitro" and "in-the-wild". The in-vitro experiments aim to thoroughly evaluate the performance and characteristics of SM, including the use of the SAT solver, on a dataset of random graphs with predefined sizes and densities. These graphs have a maximum degree of 4, which is the baseline for orthogonal drawings, and contain $20-60$ vertices, a range considered important for node-link representations in graph visualization (see, e.g., [25]). The in-the-wild experiments use the Rome Graphs dataset [20], which is one of the most popular collections of graphs [3], comprised of $11,534$ graphs, having $10-100$ vertices and $9-158$ edges, with maximum degree 13. The experiments show that for most of the typical graph drawing metrics [3], DOMUS outperforms the TSM algorithm implemented in OGDF. Moreover, SM is simple to implement: it reduces shape constraints to a SAT formula, invokes an off-the-shelf SAT solver, and computes the final drawing metrics using a slight variation of standard TSM compaction techniques. Figure 1 shows two drawings of the same graph, one produced with OGDF and the other with DOMUS.

The paper is organized as follows. Preliminaries are in Section 2. Section 3 illustrates the characterization of rectilinear drawable shaped graphs which is the milestone of SM. Section 4 describes SM, clarifying how its steps are activated and implemented and the role played by the SAT-solver. Section 5 and Section 6 show the in-vitro and the in-the-wild experiments, respectively. Conclusions and open problems are proposed in Section 7. Because of space limitations some proofs are only sketched. A full version of such proofs and other extra material is in [1].

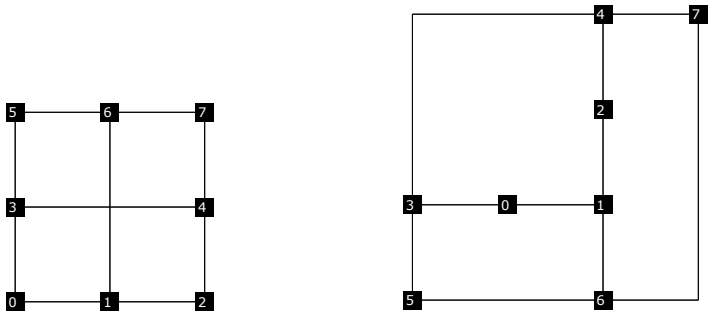


Figure 1 The graph in Fig. 2.4 of [19], represented: **(a)** with DOMUS and **(b)** with OGDF.

2 Preliminaries

For basic graph drawing terminology and definitions refer, e.g., to [19, 32].

Let G be a graph. with vertex set $V(G)$ and edge set $E(G)$. For technical reasons, we assign an arbitrary (e.g., random) orientation to each edge in $E(G)$; thus, if an edge is denoted as (u, v), it is considered as directed from u to v. However, when referring to paths, cycles, vertex degrees, or connectivity in G, we disregard these orientations. A path (resp., cycle) of G is *simple* if each of its vertices occurs only once.

A *cut-vertex* is a vertex whose removal disconnects G. Graph G is *biconnected* if it has no cut-vertex. A *biconnected component* of G is a maximal (in terms of vertices and edges) biconnected subgraph of G. A biconnected component is *trivial* if it consists of a single edge and *non-trivial* otherwise. Unless otherwise specified we assume that graphs are connected.

In a *rectilinear drawing* of G: **(i)** Each vertex $v \in V(G)$ is represented by a point with integer coordinates, denoted by $x(v)$ and $y(v)$. **(ii)** No two vertices are represented by the same point. **(iii)** Each edge is represented by either a horizontal or a vertical line segment connecting the points that represent its end-vertices. **(iv)** The interior of a segment representing an edge does not intersect any point representing a vertex. A graph is said to be *rectilinear drawable* if it admits a rectilinear drawing. Note that such a drawing requires each vertex to have degree at most 4. Hence, unless otherwise specified, we will consider graphs with maximum vertex degree 4.

We use the label set $\mathcal{L} = \{L, R, D, U\}$ to denote directions, where L, R, D, and U stand for left, right, down, and up, respectively. Also, we say that L and R are *opposite* labels, and likewise, D and U are *opposite* labels. For any label $A \in \mathcal{L}$, we denote its opposite by $\overline{A}$.

A *shape* of a graph G is a function λ that assigns to each edge $(u, v) \in E(G)$ a label from the set $\mathcal{L}$. Consider an edge (u, v). Labeling (u, v) with, say, R indicates that in the shape of G, the edge is directed to the right when traversed from u to v. Clearly, if the edge is traversed in the opposite direction (from v to u) the direction should be left. Hence, with a little abuse of notation, if $\lambda(u, v) = R$ (resp. $\lambda(u, v) = L$) we say that $\lambda(v, u) = L$ (resp. $\lambda(v, u) = R$). The other labels are treated analogously. Additionally, for each pair of edges that share a vertex v it is mandatory that $\lambda(v, u) \neq \lambda(v, w)$. This constraint ensures that two edges sharing the same vertex "cannot point in the same direction", preventing the two edges from "overlapping". We define a *shaped graph* to be a graph with a shape.

A shape can be assigned to a graph by starting from one of its rectilinear drawings, if such a drawing exists. Namely, given a rectilinear drawing Γ of a graph G we can label the edges $E(G)$ as follows. For each $(u, v) \in E(G)$: **(i)** If $x(u) < x(v)$ we set $\lambda(u, v) = R$, **(ii)** If

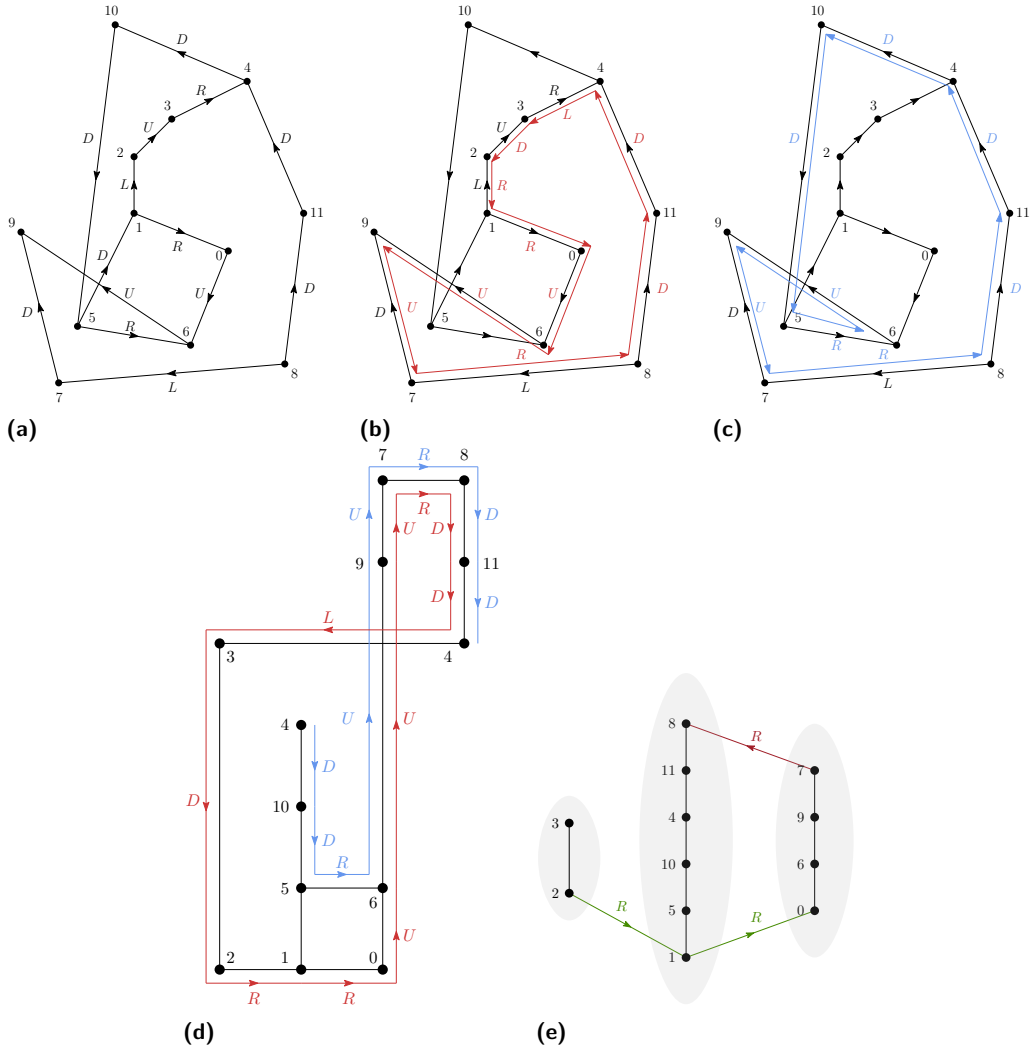


Figure 2 **(a)** A shaped graph G. **(b)** A complete cycle of G. **(c)** Another cycle of G. **(d)** A failed attempt to draw G according to its shape. **(e)** Graph G_x for the shaped graph of Figure 2a.

$x(u) > x(v)$ we set $\lambda(u, v) = L$, **(iii)** If $y(u) < y(v)$ we set $\lambda(u, v) = U$, **(iv)** If $y(u) > y(v)$ we set $\lambda(u, v) = D$. We say that this is the *shape* of Γ. It is easy to check that this labeling is consistent with the definition of shape, and hence it is also a valid shape for G.

A shaped graph is *rectilinear drawable* if it admits a rectilinear drawing having exactly its shape. The shaped graph in Figure 2a is not rectilinear drawable. Figure 2d shows a failed attempt to construct a rectilinear drawing of such a shaped graph.

Given a graph G a *subdivision* of G is a graph G' obtained by replacing certain edges of $E(G)$ with internally vertex-disjoint simple paths. The new vertices in $V(G') \setminus V(G)$ introduced along these paths are called *dummy vertices*. An *orthogonal drawing* Γ of G is a rectilinear drawing of a subdivision of G. Let v be a dummy vertex, if the two segments incident to v in Γ are one horizontal and one vertical, then v is called a *bend*.

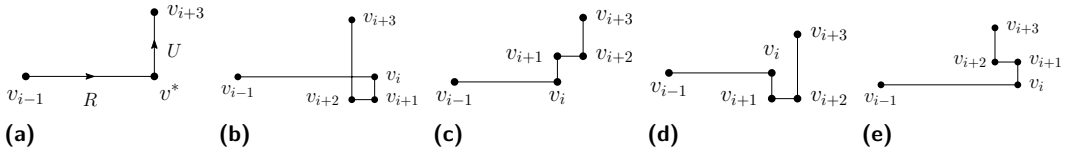


Figure 3 Illustration for the proof of Theorem 1. **(a)** The partial rectilinear drawing of c' according to the new sequence. The result of the extension of the drawing of c' to c by re-inserting labels **(b)** D and L; **(c)** U and R; **(d)** D and R; and **(e)** U and L.

3 Rectilinear Drawable Shapes and the Shape of Cycles

The methodology we present in Section 4 is based on the results presented in this section.

Consider a shaped simple cycle $c = v_0, v_1, \ldots, v_{p-1}$, with $p \geq 4$, of G. We say that c is *complete* if it contains four pairs of vertices v_i, v_{i+1}, v_j, v_{j+1}, v_h, v_{h+1}, and v_k, v_{k+1} such that: **(i)** $\lambda(v_i, v_{i+1}) = L$, **(ii)** $\lambda(v_j, v_{j+1}) = R$, **(iii)** $\lambda(v_h, v_{h+1}) = D$, **(iv)** $\lambda(v_k, v_{k+1}) = U$, where the vertices indexes of c are intended mod p. An example of a complete cycle is shown in Figure 2b, while Figure 2c illustrates a cycle that is not complete.

▶ **Theorem 1.** *A shaped simple cycle $c = v_0, v_1, \ldots, v_{p-1}$ with $p \geq 4$ is rectilinear drawable if and only if it is complete.*

Sketch of Proof. If c admits a rectilinear drawing, then intersecting it with horizontal and vertical lines (avoiding vertices) reveals edges labeled U, D, L, and R. Hence, c is complete. Conversely, suppose c is complete. We prove that it admits a rectilinear drawing by induction on the number of vertices p. For $p = 4$, the completeness implies the presence of all directions in $\{L, R, U, D\}$. The only valid labelings without consecutive opposites are R–U–L–D and R–D–L–U, which form rectangles. Now, assume that any complete cycle with $p - 2$ vertices ($p \geq 6$) is drawable. Let c be a complete cycle with $p \geq 6$ vertices. Without loss of generality, assume no two consecutive labels are equal. Then, the label sequence $\Lambda = s_0, \ldots, s_{p-1}$ alternates between $\{L, R\}$ and $\{U, D\}$, contains all four labels, and avoids consecutive equal or opposite pairs. Then, there exists a removable pair s_i, s_{i+1} such that the resulting cycle c' remains complete. By induction, c' is drawable. Denote (see Figure 3) by (v_{i-1}, v^*) (resp., by (v^*, v_{i+3})) the edge of c' corresponding to s_{i-1} (resp., s_{i+2}), where v^* represents the vertices v_i, v_{i+1} and v_{i+2} that have been merged together from the removal of the two edges. Reinserting s_i, s_{i+1} as a right-angle bend (e.g., $\langle U, R \rangle$, $\langle D, L \rangle$) and applying simple geometric adjustments yields a drawing of c. Thus, c is rectilinear drawable. ◀

To test if a shaped graph is rectilinear drawable we can use the following theorem, which has been proved in [30].

▶ **Theorem 2** ([30, Theorem 3]). *Given a shaped graph G, there exists an $O(|V(G)| \cdot |E(G)|)$ algorithm to test if it is rectilinear drawable. In the positive case a rectilinear drawing can be found in $O(|V(G)| \cdot |E(G)|)$ time.*

The proof in [30] is based on constructing, starting from the given shaped graph, two auxiliary directed graphs which we call G_x and G_y. For the sake of readability, we call the vertices and the edges of such graphs *nodes* and *arcs*, respectively.

We say that two vertices v_0 and v_{k-1} of G are *x-aligned* if a path $v_0, \ldots, v_{k-1}$ exists in G such that $\lambda(v_i, v_{i+1}) = D$ $(i = 0, \ldots, k-2)$. Also, we say that a vertex of G is *x-aligned* with itself. Graph G_x is defined as follows. A node of G_x is a maximal set of x-aligned vertices of G. Given two nodes μ and ν of G_x, by the definition of shape, they are disjoint sets. There

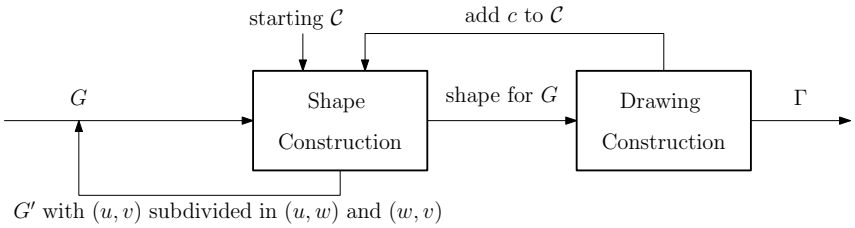


Figure 4 SM: a methodology for constructing orthogonal drawings of graphs.

is an arc from node μ to node ν if there are two vertices u and v of G such that $u \in \mu$, $v \in \nu$, and either $(u, v) \in E$, and $\lambda(u, v) = R$ or $(v, u) \in E$, and $\lambda(v, u) = L$. Figure 2e shows graph G_x for the shaped graph in Figure 2a. Graph G_y is defined analogously.

In [30] it is proved that if there is an arc in G_x (resp., G_y) from μ to ν, then in any rectilinear drawing of G the vertices in μ should be placed to the left of (resp., above) the vertices in ν. Also all the vertices in the same node should be vertically (resp., horizontally) aligned. This implies that a necessary condition for the shape to be rectilinear drawable is that both G_x and G_y are acyclic. Observe that graph G_x of Figure 2e is cyclic and the shaped graph in Figure 2a is not rectilinear drawable. The authors show that this condition is also sufficient by constructing a drawing. Namely, they assign the x-coordinates to the vertices of $V(G)$ as follows: (1) the same x coordinate is assigned to all the vertices corresponding to the same node of G_x, (2) different coordinates are assigned to vertices corresponding to different nodes of G_x, and (3) x-coordinates increase according to the orientations of the nodes of G_x. The y-coordinates are assigned analogously. Since in our case the vertices of $V(G)$ have degree at most 4, it easy to see that the construction of G_x and G_y, the acyclicity test, and, in case, the construction of the drawing can be accomplished in $O(|V(G)|)$ time.

We are now able to prove the following characterization.

▶ **Theorem 3.** *A shaped graph G is rectilinear drawable if and only if all its simple cycles are complete. In case G is not rectilinear drawable, a simple cycle of G which is not complete can be found in $O(|V(G)|)$ time.*

Sketch of Proof. The necessity follows from Theorem 1. To prove the sufficiency consider a shaped graph G and its auxiliary directed graphs G_x and G_y. From the proof of Theorem 2 we have that the presence of a cycle in G_x or G_y implies that G is not rectilinear drawable using the prescribed shape. We prove that a cycle in G_x or in G_y implies a non-complete simple cycle in G. Let $C_x = \mu_0, \ldots, \mu_{p-1}$ be a cycle in G_x, possibly p=1. See Figure 2e, where $p = 2$. Consider the p pairs u_i, v_{i+1} of vertices of G, such that $u_i \in \mu_i$ and $v_{i+1} \in \mu_{i+1}$, $(u_i, v_{i+1}) \in E(G)$, and $\lambda(u_i, v_{i+1}) = R$, for $i \in \{0, \ldots, p-1\}$ (where indices are taken mod p). By the construction of G_x, there exists a path Π_i in G from v_i to u_i that contains edges all labeled D or all labeled U. The concatenation of the paths and edges $\Pi_0, (u_0, v_1), \Pi_1, \ldots, (u_{p-2}, v_{p-1}), \Pi_{p-1}, (u_{p-1}, v_0)$ forms a simple cycle c in G that is not complete. The cycle of Figure 2e contains vertices 8, 11, 4, 10, 5, 1, 0, 6, 9, and 7. The construction of G_x and of G_y, the acyclicity test, the search of a cycle C_x, and the computation of c can all be done in $O(|V(G)|)$ time. ◀

4 A Methodology for Constructing Orthogonal Drawings of Graphs

This section presents SM. Given an n-vertex graph G as input, it produces an orthogonal drawing Γ of G as output. SM consists of two steps, called *Shape Construction* and *Drawing Construction*, see Figure 4. Each of these steps can be repeated multiple times.

4.1 Shape Construction

We know from Theorem 3 that G admits a rectilinear drawable shape if and only if it admits a shape in which all simple cycles are complete. Consequently, the Shape Construction step may attempt to compute a shape for G that enforces completeness for all simple cycles. However, since the number of simple cycles in G can be exponential in the size of G, checking all of them is computationally infeasible. To address this, the first phase of Shape Construction, called *Cycle Selection*, selects a subset $\mathcal{C}$ of simple cycles in G so that the Shape Construction searches for a shape that ensures completeness only for the cycles in $\mathcal{C}$, with the hope that this will be enough to enforce completeness for all cycles.

If no such shape exists, this implies that a rectilinear drawable shape for G does not exist–since adding more constraints cannot make a previously unsolvable problem solvable. In this case, the Shape Construction step attempts to identify an edge (u, v) that is "responsible" for the infeasibility, and splits it into two edges, (u, w) and (w, v), by introducing a dummy vertex w. It then restarts the process on the modified graph G', which is a subdivision of G. The vertex w will possibly be drawn as a bend in the final orthogonal drawing. Additionally, splitting the edge (u, v) indirectly alters the cycle set $\mathcal{C}$, producing a new set $\mathcal{C}'$, as the new vertex w is now included in all cycles of $\mathcal{C}$ that originally contained (u, v).

If such a shape exists, this does not necessarily imply that it is rectilinear drawable. Thus, the shape is passed as input to the Drawing Construction step for further verifications.

4.2 Drawing Construction

The Drawing Construction step relies on two theorems. Theorem 2, states that a shape can be efficiently tested for rectilinear drawability, and if the test is positive, a rectilinear drawing can be constructed efficiently. Theorem 3, states that if a shape is not rectilinear drawable a cycle c that is not complete can be efficiently identified. Hence, the Drawing Construction step either produces Γ directly or suggests to the Shape Construction step to add c to the cycle set $\mathcal{C}$, selectively enlarging the set of cycles considered for completeness.

Observe that, in general, due to the possible introduction of dummy vertices during the Shape Construction step, the resulting rectilinear drawing Γ is the rectilinear drawing of a subdivision of G. Hence, Γ is an orthogonal drawing of G.

4.3 Implementing the Shape and the Drawing Construction

Implementing the Shape Construction step essentially involves selecting an initial cycle set $\mathcal{C}$ and checking whether a shape of G exists that enforces completeness for all cycles in $\mathcal{C}$.

The selection of $\mathcal{C}$ can be done in various ways. A natural and intuitive strategy is to include cycles that cover all edges belonging to non-trivial biconnected components of G. Further details are given in Section 5.

The checking for the existence of a shape is the most important part of the Shape Construction. We do this by converting completeness constraints into a CNF Boolean formula, $\mathcal{F}_{G,\mathcal{C}}$ and then by using a SAT solver to check satisfiability. This method benefits from (1) the efficiency of modern SAT solvers and (2) the solver's ability to provide a proof if unsatisfiable. From this proof, we identify an edge label variable causing unsatisfiability, indicating the edge (u, v) is over constrained, so we split it by adding a dummy vertex.

Formula $\mathcal{F}_{G,\mathcal{C}}$ has four boolean variables for each edge (v, w) of $E(G)$. Such variables state if $\lambda(v, w)$ is equal to L, R, D, or U. It also has three types of clauses to state that: (1) each edge is assigned to exactly one label, (2) no vertex has two adjacent edges with the same direction, and (3) each simple cycle in $\mathcal{C}$ is complete.

The full formal description of $\mathcal{F}_{G,\mathcal{C}}$ is as follows. Formula $\mathcal{F}_{G,\mathcal{C}}$ has four boolean variables for each edge (v, w) of $E(G)$. We call them $\ell_{v,w}$, $r_{v,w}$, $d_{v,w}$, and $u_{v,w}$. We have that:

1. $\ell_{v,w} = true$ iff $\lambda(v, w) = L$,
2. $r_{v,w} = true$ iff $\lambda(v, w) = R$,
3. $d_{v,w} = true$ iff $\lambda(v, w) = D$, and
4. $u_{v,w} = true$ iff $\lambda(v, w) = U$

To simplify the description, we also introduce the variable $\ell_{w,v}$ (resp. $r_{w,v}$), which is equal to $r_{v,w}$ (resp. $\ell_{v,w}$). Analogously, we introduce the variable $d_{w,v}$ (resp. $u_{w,v}$), which is equal to $u_{v,w}$ (resp. $d_{v,w}$).

Formula $\mathcal{F}_{G,\mathcal{C}}$ has three sets of clauses.

The first set of clauses ensures that each edge (v, w) is assigned exactly one label. Namely, for each edge (v, w) we have the following clauses: **(i)** $(\ell_{v,w} \vee r_{v,w} \vee d_{v,w} \vee u_{v,w})$, **(ii)** $(\neg\ell_{v,w} \vee \neg r_{v,w})$, **(iii)** $(\neg d_{v,w} \vee \neg\ell_{v,w})$, **(iv)** $(\neg d_{v,w} \vee \neg r_{v,w})$, **(v)** $(\neg u_{v,w} \vee \neg\ell_{v,w})$, **(vi)** $(\neg u_{v,w} \vee \neg r_{v,w})$, **(vii)** $(\neg u_{v,w} \vee \neg d_{v,w})$. Clause **(i)** is true if at least one label is assigned to (v, w), clauses **(ii)**–**(vii)** are true if at most one label is assigned to (v, w).

The second set of clauses guarantees that for each vertex v with at least two neighbors, there are no two of such neighbors w' and w'' of v such that $\lambda(v, w') = \lambda(v, w'')$. Let $u_0, \ldots, u_k$ be the neighbors of v, with $k = 1, 2, 3$ according to the degree of v minus one. We distinguish two cases. If v has degree 4, we have 4 clauses, $(\ell_{v,u_0} \vee \ell_{v,u_1} \vee \ell_{v,u_2} \vee \ell_{v,u_3})$, $(r_{v,u_0} \vee r_{v,u_1} \vee r_{v,u_2} \vee r_{v,u_3})$, $(d_{v,u_0} \vee d_{v,u_1} \vee d_{v,u_2} \vee d_{v,u_3})$, and $(u_{v,u_0} \vee u_{v,u_1} \vee u_{v,u_2} \vee u_{v,u_3})$, that are satisfied if for each label $l \in \mathcal{L}$ there exists an index i $(i = 0, \ldots, 3)$ such that $\lambda(v, u_i) = l$. Otherwise (v has degree 2 or 3), for each pair of neighbors u_i and u_j of v, we have $(\neg\ell_{v,u_i} \vee \neg\ell_{v,u_j})$, $(\neg r_{v,u_i} \vee \neg r_{v,u_j})$, $(\neg d_{v,u_i} \vee \neg d_{v,u_j})$, and $(\neg u_{v,u_i} \vee \neg u_{v,u_j})$.

The third set of clauses guarantees that all simple cycles in $\mathcal{C}$ are complete. For each simple cycle $v_0, v_1, \ldots, v_{n-1}$ of $\mathcal{C}$ we impose that it contains all labels in $\mathcal{L}$ while traversing it. This is done with four clauses as follows: $(\ell_{v_0,v_1} \vee \ell_{v_1,v_2} \vee \cdots \vee \ell_{v_{n-2},v_{n-1}} \vee \ell_{v_{n-1},v_0})$, $(r_{v_0,v_1} \vee r_{v_1,v_2} \vee \cdots \vee r_{v_{n-2},v_{n-1}} \vee r_{v_{n-1},v_0})$, $(d_{v_0,v_1} \vee d_{v_1,v_2} \vee \cdots \vee d_{v_{n-2},v_{n-1}} \vee d_{v_{n-1},v_0})$, and $(u_{v_0,v_1} \vee u_{v_1,v_2} \vee \cdots \vee u_{v_{n-2},v_{n-1}} \vee u_{v_{n-1},v_0})$

It is easy to see that $\mathcal{F}_{G,\mathcal{C}}$ is satisfiable if and only if G has a shape such that all the cycles in $\mathcal{C}$ are complete.

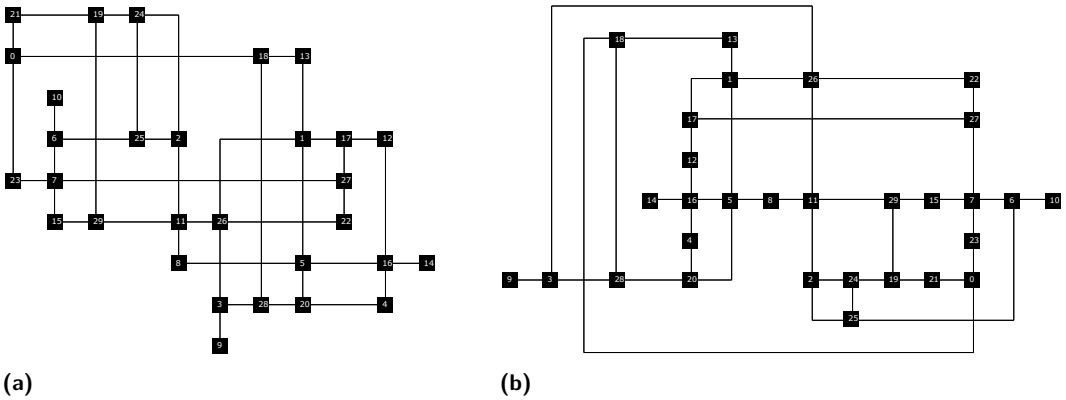
Implementing the Drawing Construction step primarily involves building the two auxiliary graphs G_x and G_y and checking each for acyclicity. If either graph contains a cycle, it can be used to identify a corresponding cycle c in G that is not complete.

Finally, we observe that the process described in the methodology is guaranteed to terminate, since an orthogonal drawing of a graph always exists (see, e.g, [10, 11]).

5 An Experimental Evaluation on Maximum Degree 4 Graphs

We implemented SM in our tool DOMUS and evaluate its effectiveness through comprehensive "in vitro" experiments conducted using DOMUS. Specifically, we describe: **(i)** the implementation choices in DOMUS that define some aspects of SM, **(ii)** the benchmark adversary for comparison, **(iii)** the graph dataset and its key characteristics, **(iv)** the computational platform used, **(v)** the evaluation metrics, **(vi)** the experimental results for each metric, and **(vii)** internal performance indicators, such as SAT solver invocations per graph.

Two drawings of a graph in the dataset are in Figure 5. **(i)** In DOMUS, we made specific implementation choices regarding the SAT solver, the initial cycle set $\mathcal{C}$, coordinate computation, and randomization. For the SAT solver, we chose Glucose [2] because it is open source and it is well known for its efficiency. Importantly, Glucose provides proofs of



(a) **(b)**

Figure 5 A graph in the dataset: **(a)** drawn by DOMUS and **(b)** drawn by OGDF.

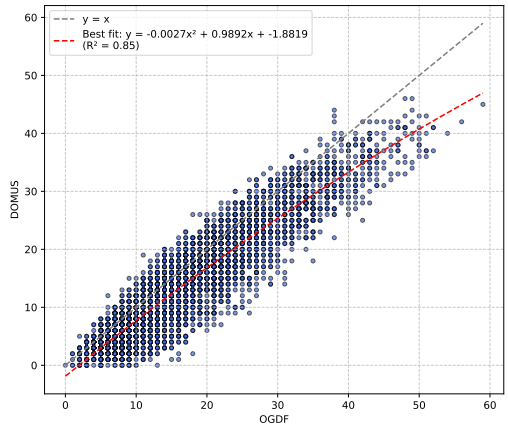
unsatisfiability, which we use to identify edges to split, thereby increasing flexibility in the drawing. To initialize the cycle set $\mathcal{C}$, DOMUS computes a *cycle basis* covering all edges in the non-trivial biconnected components of the input graph G. This is done via a BFS from an arbitrary root to construct a tree T; for each non-tree edge (u, v), we add the cycle formed by (u, v) and the unique paths from u and v to their lowest common ancestor in T. This results in an initial set of size $|\mathcal{C}| = |E(G)| - |V(G)| + 1$. For coordinate assignment, after computing G_x and G_y, DOMUS uses a compaction algorithm similar to that in OGDF. For details on compaction algorithms see, e.g., [6, 7, 29, 35]. Finally, Glucose can make random choices. In order to enforce replicability of the experiments we forced it to work deterministically.

(ii) For the motivations discussed in Section 1, we compared the performance of DOMUS against the TSM implementation available in OGDF [13]. Another option was to use as a benchmark one of the tools (see, e.g. [28]) that have been evaluated via user studies. However, this comparison would be useless, since most of this tools adopt a definition of orthogonal drawing which is different from the widely adopted one given in Section 2. E.g., even if a vertex has degree less or equal than 4 more than one of its incident edges can exit from the same direction. Also, vertices and bends neither have integer coordinates nor it is easy to "snap" them to a grid (see [1] for a comparison with these models).

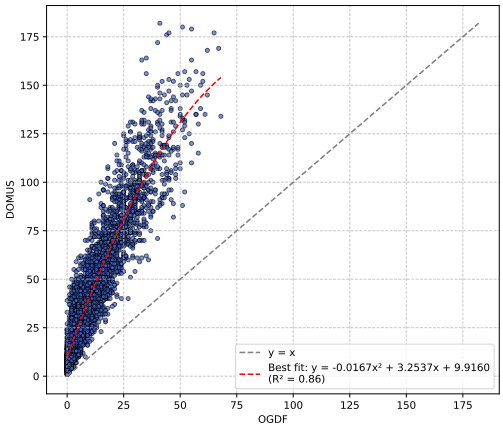
(iii) We generated 4,100 connected graphs uniformly at random, each with maximum vertex degree 4. Their sizes range from 20 to 60 vertices, and their densities – defined as the ratio between the number of edges to the number of vertices – span 1.25 to 1.75 in steps of 0.005. For degree 4 graphs, the theoretical maximum density is 2. Namely, for each $n = 20, \ldots, 60$ and $i = 1, 2, \ldots, 100$ we generated a graph $G_{n,i}$ with density $d_i = 1.25 + i \cdot (1.75 - 1.25)/100$. Hence, for each value of n the graphs have roughly 100 possible densities ranging from 1.25 to 1.75, in increments of 0.005. In terms of number of edges we have $|E(G_{n,i})| = \lfloor n \cdot d_i \rfloor$. For instance, the graph $G_{20,50}$ has 30 edges, while $G_{60,100}$ has 105 edges. For each n and d_i we initialized a graph with n vertices and no edges. Then, we repeatedly randomized two vertices u and v of the graph and added an edge between them, until d_i was reached. If $u = v$ or if (u, v) was already in the graph, or if either u or v was already degree 4, we skipped the pair. Finally, the instance was rejected if non-connected.

Conducting experiments on a set of randomly generated graphs was a necessary choice, as, to the best of our knowledge, no publicly available collection of maximum-degree-4 graphs is currently recognized as a standard benchmark for graph drawing experiments.

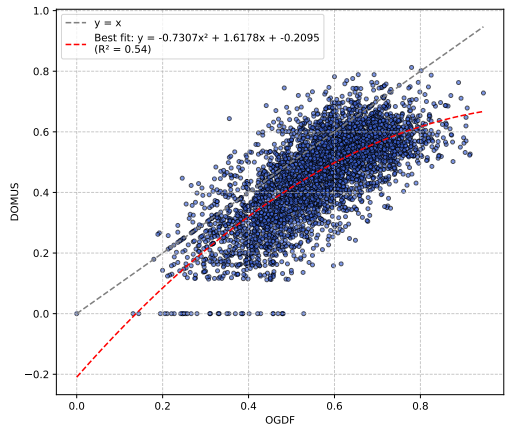
(iv) All the experiments were performed running OGDF and DOMUS on a personal computer with a 3.15 GHz Intel processor (comparable with an M1 Apple processor).



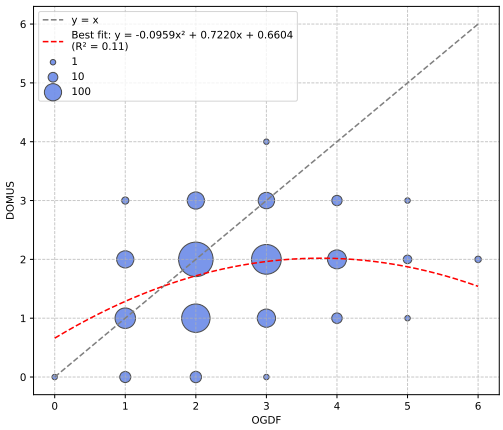
(a) Bends.



(b) Crossings.



(c) Bends Deviation.

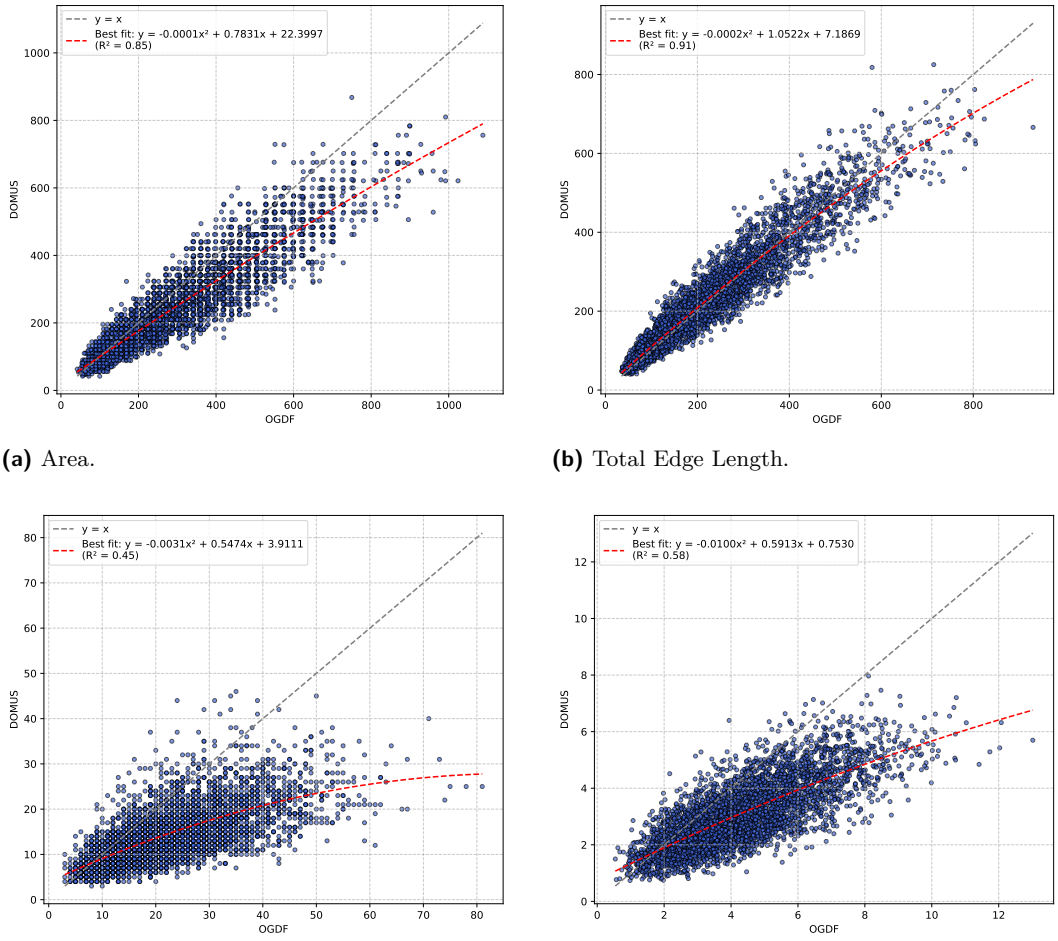


(d) Max Bends.

Figure 6 Effectiveness of OGDF and DOMUS "in vitro": Bends and Crossings.

(v) Following the framework proposed in [20], we compared the drawings produced by DOMUS and those generated by OGDF using the following metrics: the total number of **Bends** in the drawing; the total number of edge **Crossings**; the standard deviation of the number of bends per edge (**Bends Deviation**); the maximum number of bends on any single edge (**Max Bends**); the **Area** occupied by the drawing; the sum of the length of all the edges (**Total Edge Length**); the length of the longest edge (**Max Edge Length**); the standard deviation of edge lengths (**Edge Length Deviation**) and the total computation **Time** to build the drawing, measured in seconds. Using the terminology in [3], we adopt most of the metrics suggested for "quantitative individual" and "aggregated" evaluations. A note on how the above metrics were computed is in [1].

(vi) The results of our experiments are shown in Figures 6 and 7, where each random graph is represented by a small circle. The x-coordinate corresponds to the metric value produced by OGDF, while the y-coordinate reflects the value obtained by DOMUS. Points below the bisector of the first quadrant (gray dashed line) indicate cases where DOMUS outperformed OGDF; points above indicate the opposite. To account for overlapping data points (caused by graphs with same metric values), each plot includes a red dashed trend line summarizing the comparative performance. We also report the coefficient of determination (R^2, with $0 \leq R^2 \leq 1$), with values closer to 1 indicating a better interpolation.



(a) Area.

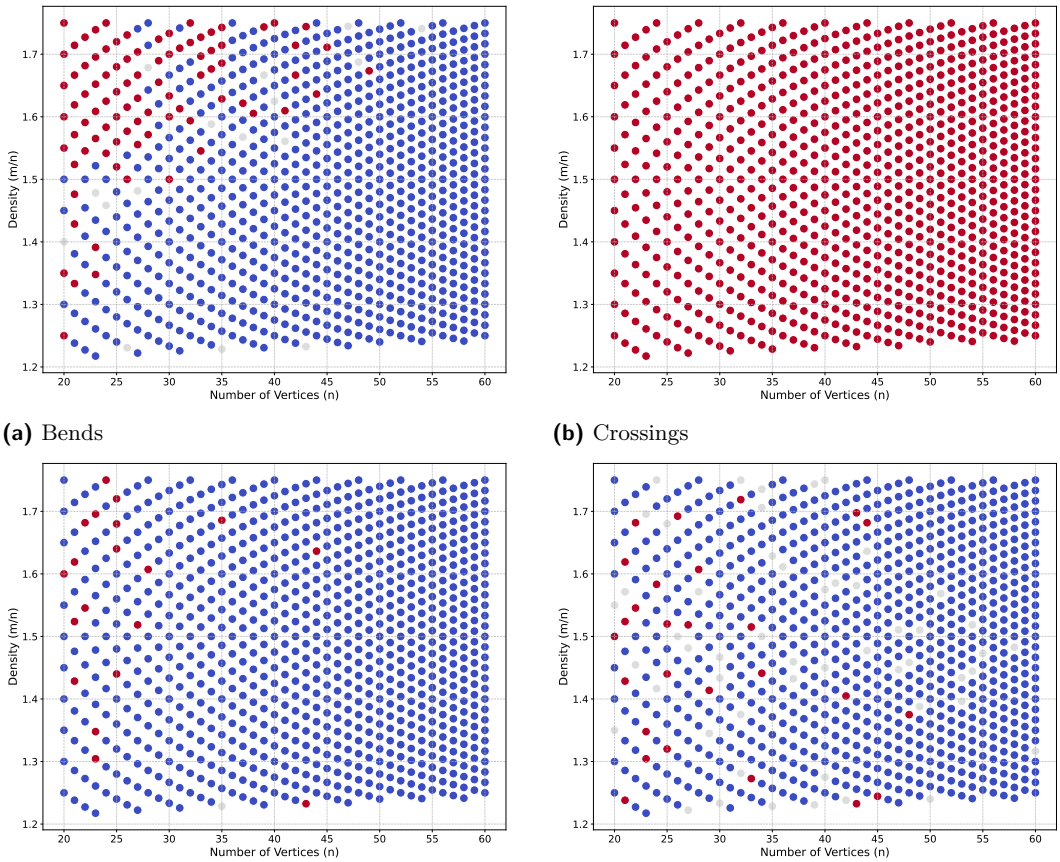
(b) Total Edge Length.

(c) Max Edge Length.

(d) Edge Length Deviation.

Figure 7 Effectiveness of OGDF and DOMUS "in vitro": Area and Edge Length.

Figure 6a focuses on Bends. DOMUS outperformed OGDF on 78% of the graphs and matched its performance on another 7%. The trend line indicates a modest quadratic improvement, a linear improvement of approximately 2%, and a constant improvement of about 2.5 bends per graph. Notably, 89 of the random graphs admit a rectilinear drawing. Conversely, as expected (Figure 6b), OGDF sharply outperforms DOMUS in terms of Crossings. In this case, 149 random graphs admit a planar drawing. Additional insights regarding bends are in Figures 6c and 6d, which report the Bends Deviation and the Max Bends per edge, respectively. For the former metric, DOMUS outperforms OGDF on 85% of the graphs. For the latter, DOMUS matches OGDF on 50% of the graphs and outperforms it in the remaining 50%. Since the possible values of the maximum number of bends per edge are relatively few, many graphs correspond to the same point. Because of this, in Figure 6d we represent with a variable size circle the number of graphs having the maximum number of bends per edge. The area of each circle is proportional to the corresponding number of graphs. These results indicate that our approach not only reduces the total number of bends but also achieves a more uniform distribution of bends across edges, thereby avoiding drawings with edges that contain long sequences of consecutive bends.



(a) Bends

(b) Crossings

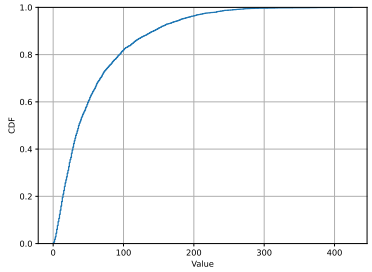
(c) Bends Deviation

(d) Max Bends

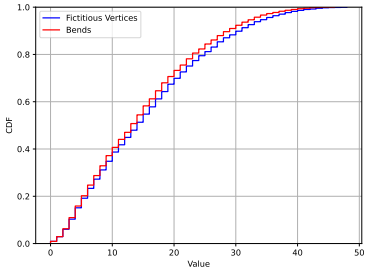
Figure 8 The effect of density: Bends and Crossings. Blue: DOMUS is better. Red: OGDF is better. Grey: parity.

DOMUS yields more compact drawings compared to those generated by OGDF (Figure 7a). Specifically, our approach results in smaller drawing areas for 78% of the graphs, and matches OGDF in 18% of the cases. The trend line indicates a consistent improvement, with an average area reduction of approximately 25%. We also achieve slightly better results than OGDF in terms of edge length. DOMUS produces drawings with lower Total Edge Length for 50% of the graphs and matches OGDF on 1% of them (Figure 7b). More significantly, DOMUS outperforms OGDF on Max Edge Length (Figure 7c) for 87% of the graphs and obtain equal results for an additional 10%. Similarly, for Edge Length Deviation (Figure 7d), DOMUS performs better on 89% of the graphs. These results indicate that SM not only tends to produce slightly shorter edges overall but also ensures a more uniform edge length distribution, thereby avoiding disproportionately long edges that can hinder readability and spatial coherence.

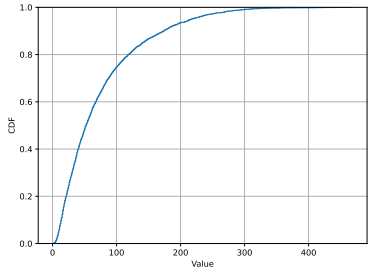
Figure 8 illustrates how the relative performance of DOMUS and OGDF varies with the number of vertices and the density of the input graphs. In each subfigure, a circle positioned at coordinates (x, y) represents all graphs with x vertices and density y. For a given metric M and each pair (x, y), let M_O (respectively, M_D) denote the average value of M for the drawings produced by OGDF (respectively, DOMUS) over all graphs with x vertices and density y. The circle at (x, y) is colored red, blue, or gray if $M_O < M_D$, $M_O > M_D$, or $M_O = M_D$,



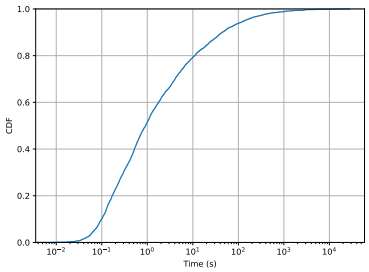
(a) Cycles added during the computation.



(b) Dummy vertices and bends.



(c) SAT invocations.



(d) Time.

Figure 9 Several aspects of the DOMUS computations.

respectively. Figure 8a shows that for small values of n, increasing graph density tends to favor OGDF. However, this advantage gradually diminishes as n increases. For the Area, we have a similar trend. The other metrics do not show any clear dependency on either the number of vertices or the density (Figures 8b–8d). [1] contains additional diagrams about the density.

(vii) To further characterize the computations performed by DOMUS, for each run on a single graph, we measured the following (see Figure 9).

First, we measured how many cycles were added to $\mathcal{C}$ during the computation. Each addition corresponds to a case where the current shape was found to be non-rectilinear drawable, triggering a new invocation of Shape Construction from Drawing Construction. In each of such invocations, the SAT solver was able to quickly find a satisfying assignment. The total number of Drawing Construction calls equals this number plus one. Figure 9a shows a cumulative distribution function (CDF) of these data. For instance, in 80% of the graphs, no more than 100 cycles were added to $\mathcal{C}$.

Second, we measured the number of dummy vertices introduced by the Shape Construction, corresponding to how often the Shape Construction called itself due to the absence of a shape satisfying the constraints. In each of these invocations, the formula given to the SAT solver was unsatisfiable. Figure 9b shows that at most 25 dummy vertices have been added for 80% of graphs. Such vertices do not necessarily become bends in the final drawing, since some of them can get an angle of π between their incident edges. So, a dummy vertex that does not become a bend can be viewed as a failure in the attempt to decrease the constraints of the drawing. Hence, we also computed how many dummy vertices actually became bends in the final drawing. This is an indirect measure of the effectiveness of using SAT unsatisfiability proofs to introduce bends. Figure 9b shows that almost all the dummy vertices became bends in the drawing.

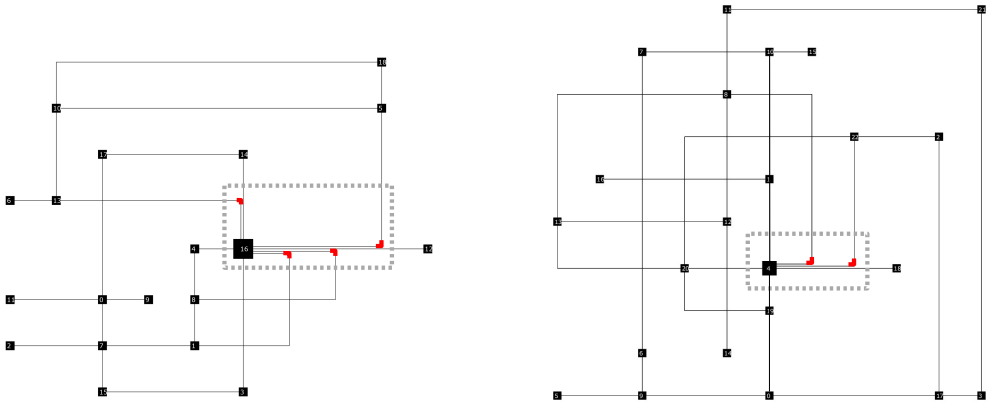


Figure 10 Illustrations of how DOMUS manages a vertex with degree greater than 4. The dashed boxes show the expansions and the red segments show the bends that are added to the drawing.

Third, we measured the total number of SAT solver invocations, computed as the sum of the Shape Construction self-invocations, the Shape Construction calls by the Drawing Construction, plus one. Figure 9c shows that at most 120 SAT invocations have been done for 80% of graphs. The similarity between Figure 9a and Figure 9c shows that most of the SAT solver calls (80%) depend on the need to add a cycle to $\mathcal{C}$. Also, the average number of Boolean variables and clauses in the input to the SAT solver were 352 and 1224, respectively.

Fourth, Figure 9d shows that computing a drawing for DOMUS required at most 10 seconds for 80% of graphs and that there are graphs that required more than 100 seconds, with an average latency of 70 seconds. OGDF did its work in an average of 0.06 seconds.

6 An Experimental Evaluation on the Rome Graphs

With small modifications, it is possible to apply our methodology to construct orthogonal drawings of graphs with degree greater than 4. In representing such graphs, we use the same convention introduced in [24]. Namely, the edges exiting the same side of a vertex can partially overlap (they are distanced by a very small amount) only for the first segment representing them. See Figure 10. The variation of the methodology works as follows. First, we modify the $\mathcal{F}_{G,\mathcal{C}}$ formula in such a way that if a vertex v has degree greater than 4 more than one edge can enter v from the same direction, while keeping the constraint that at least one edge enters v from all the directions. After that, all the machinery that looks for a shape, including the usage of the SAT solver, stays the same. Second, once a shape has been found, v is temporarily expanded into a box (which will not be shown in the drawing) and the edges incident to the same side of v are separated by introducing inside the box dummy vertices that become bends in the final drawing. After that, a metric is computed with the same technique as before. See drawings of Rome graphs in Figure 10.

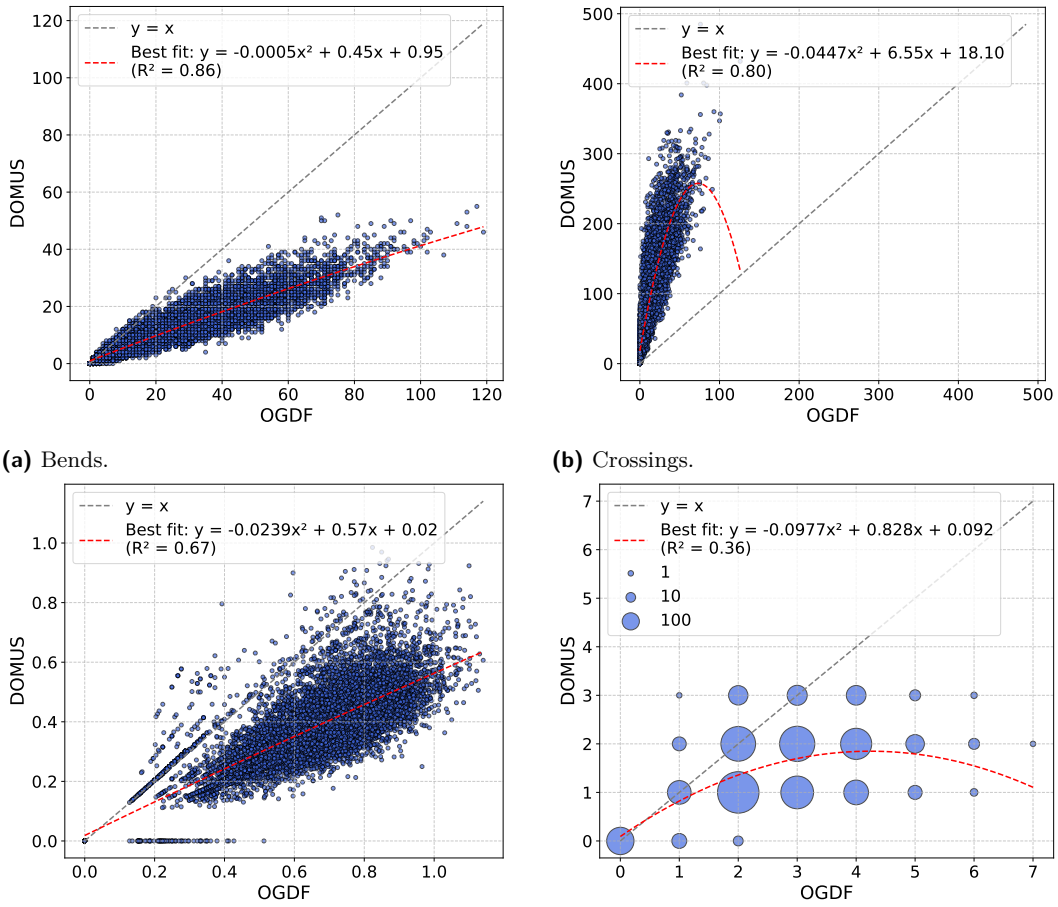
We compared OGDF and DOMUS in experiments that we call "in the wild" using the Rome Graphs dataset [20], which was originated by real-life applications and whose vertices have no degree restriction (see Figures 11 and 12). The experiments further increase the advantage of DOMUS with respect to OGDF (see Section 5) in terms of Bends (Figures 11a, 11c, and 11d) and confirm that OGDF performs much better in terms of Crossings (Figure 11b). This probably depends on the low densities of the Rome Graphs, which are close to planar in

terms of graph edit distance. Further, DOMUS performs better than OGDF in terms of Area, Max Edge Length, and Edge Length Deviation (Figures 12a, 12c, and 12d) and the two tools are comparable for Total Edge Length (Figure 12b). Hence, the presence of vertices of degree greater than 4 does not change too much the experimental results. OGDF was, even in this case, much faster than DOMUS, since the average duration of its computations was 0.07 seconds with a maximum of 3.3 seconds. The DOMUS computations took an average of 1.44 seconds, with a maximum of 769 seconds.

7 Conclusions and Open Problems

About forty years after the introduction of the Topology-Shape-Metrics (TSM) approach for orthogonal drawings, we propose a novel methodology that subverts the TSM pipeline by prioritizing bend minimization over crossing minimization. Our experimental results demonstrate that this shift yields improvements across most standard metrics used in Graph Drawing to evaluate the effectiveness of a layout algorithm.

As a final remark, we observe that our methodology can be exploited to construct drawings incorporating several types of constraints (for an introduction to constrained orthogonal drawings, see, e.g., [23, 41]). If the graph contains edges that should be drawn upward it is



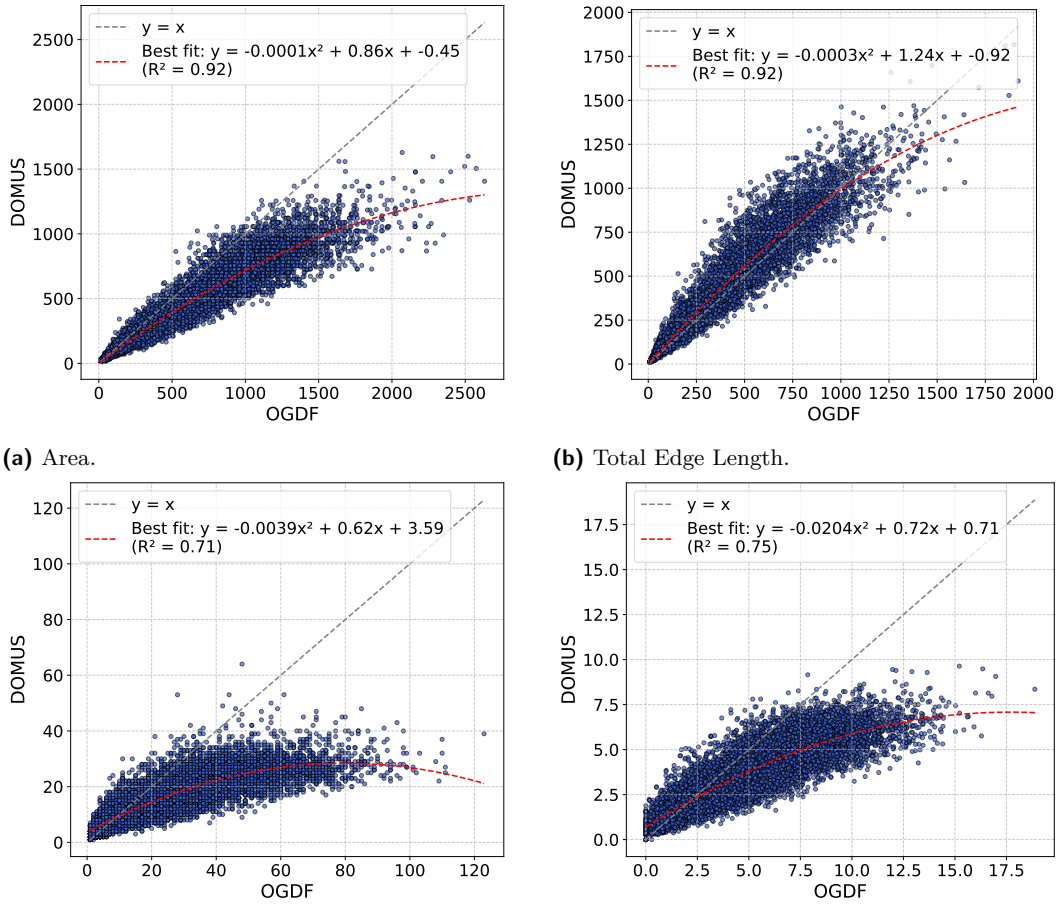
(a) Bends.

(b) Crossings.

(c) Bends Dev.

(d) Max Bends.

Figure 11 Effectiveness of OGDF and DOMUS "in the wild": Bends and Crossings.



(a) Area.

(b) Total Edge Length.

(c) Max Edge Length.

(d) Edge Length Deviation.

Figure 12 Effectiveness of OGDF and DOMUS "in the wild": Area and Edge Length.

easy to impose this by setting the value true for the corresponding u-variables of the $\mathcal{F}_{G,\mathcal{C}}$ formula. Similarly, giving the appropriate true-false values to the corresponding ℓ-, r-, d-, and u-variables it is possible to specify that certain edges should enter their end-vertices respecting some specific orientations (so called side-constraints). Further, it is easy to specify that a given edge should be horizontal or vertical or to impose the shape of entire sub-graphs.

We open several promising research directions, including the following:

- Does there exist, at least for some families of graphs, a "small" set of cycles whose completeness implies the completeness of all cycles in G? If so, can such a set be efficiently computed? In [1] it is shown that this problem is not trivial.
- Our methodology entirely disregards crossings. Is it possible to modify the approach to incorporate constraints that upper-bound the number of crossings?
- TSM and our methodology can be viewed as two extremes of a broader design space for constructing orthogonal drawings. Can hybrid methods be developed that combine elements of both approaches?
- In this paper, we employed compaction techniques originally developed within the TSM paradigm. It would be interesting to design compaction methods that do not work on the planarized underlying graph, but that only preserve the shape of the edges.

- Our approach can be leveraged to compute a maximal rectilinear drawable subgraph by iteratively invoking a SAT solver. Can the methodology be redesigned or optimized for greater efficiency when solving this specific problem?

For the sake of reproducibility all our software and our data sets are publicly available at `https://github.com/shape-metrics/domus`.

References

1 Giordano Andreola, Susanna Caroppo, Giuseppe Di Battista, Fabrizio Grosso, Maurizio Patrignani, and Allegra Strippoli. A walk on the wild side: a shape-first methodology for orthogonal drawings, 2025. arXiv:2508.19416.

2 Gilles Audemard and Laurent Simon. On the glucose SAT solver. *Int. J. Artif. Intell. Tools*, 27(1):1840001:1–1840001:25, 2018. doi:10.1142/S0218213018400018.

3 Sara Di Bartolomeo, Tarik Crnovrsanin, David Saffo, Eduardo Puerta, Connor Wilson, and Cody Dunne. Evaluating graph layout algorithms: A systematic review of methods and best practices. *Comput. Graph. Forum*, 43(6), 2024. doi:10.1111/CGF.15073.

4 Carlo Batini, Enrico Nardelli, and Roberto Tamassia. A layout algorithm for data flow diagrams. *IEEE Trans. Software Eng.*, 12(4):538–546, 1986. doi:10.1109/TSE.1986.6312901.

5 Carlo Batini, Maurizio Talamo, and Roberto Tamassia. Computer aided layout of entity relationship diagrams. *J. Syst. Softw.*, 4(2-3):163–173, 1984. doi:10.1016/0164-1212(84)90006-2.

6 Michael A. Bekos, Carla Binucci, Giuseppe Di Battista, Walter Didimo, Martin Gronemann, Karsten Klein, Maurizio Patrignani, and Ignaz Rutter. On turn-regular orthogonal representations. In David Auber and Pavel Valtr, editors, *Graph Drawing and Network Visualization - GD 2020*, volume 12590 of *LNCS*, pages 250–264. Springer, 2020. doi:10.1007/978-3-030-68766-3_20.

7 Michael A. Bekos, Carla Binucci, Giuseppe Di Battista, Walter Didimo, Martin Gronemann, Karsten Klein, Maurizio Patrignani, and Ignaz Rutter. On turn-regular orthogonal representations. *J. Graph Algorithms Appl.*, 26(3):285–306, 2022. doi:10.7155/JGAA.00595.

8 Michael A. Bekos, Michael Kaufmann, and Christian Zielke. The book embedding problem from a sat-solving perspective. In Emilio Di Giacomo and Anna Lubiw, editors, *Graph Drawing and Network Visualization - GD 2015*, volume 9411 of *LNCS*, pages 125–138. Springer, 2015. doi:10.1007/978-3-319-27261-0_11.

9 Therese Biedl, Thomas Bläsius, Benjamin Niedermann, Martin Nöllenburg, Roman Prutkin, and Ignaz Rutter. Using ILP/SAT to determine pathwidth, visibility representations, and other grid-based graph drawings. In Stephen K. Wismath and Alexander Wolff, editors, *Graph Drawing - 21st International Symposium, GD 2013*, volume 8242 of *LNCS*, pages 460–471. Springer, 2013. doi:10.1007/978-3-319-03841-4_40.

10 Therese Biedl and Goos Kant. A better heuristic for orthogonal graph drawings. In Jan van Leeuwen, editor, *Algorithms — ESA '94*, pages 24–35, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg. doi:10.1007/BFb0049394.

11 Therese C. Biedl and Goos Kant. A better heuristic for orthogonal graph drawings. *Comput. Geom.*, 9(3):159–180, 1998. doi:10.1016/S0925-7721(97)00026-6.

12 Timo Brand, Daniel Faber, Stephan Held, and Petra Mutzel. A customized sat-based solver for graph coloring. *CoRR*, abs/2504.04821, 2025. doi:10.48550/arXiv.2504.04821.

13 Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W Klau, Karsten Klein, Petra Mutzel, et al. The Open Graph Drawing Framework (OGDF). *Handbook of graph drawing and visualization*, 2011:543–569, 2013.

14 Markus Chimani and Robert Zeranski. Upward planarity testing via SAT. In Walter Didimo and Maurizio Patrignani, editors, *Graph Drawing - GD 2012*, volume 7704 of *LNCS*, pages 248–259. Springer, 2012. doi:10.1007/978-3-642-36763-2_22.

15 Markus Chimani and Robert Zeranski. Upward planarity testing: A computational study. In Stephen K. Wismath and Alexander Wolff, editors, *Graph Drawing - GD 2013*, volume 8242 of *LNCS*, pages 13–24. Springer, 2013. doi:10.1007/978-3-319-03841-4_2.

16 Markus Chimani and Robert Zeranski. Upward planarity testing in practice: SAT formulations and comparative study. *ACM J. Exp. Algorithmics*, 20:1.2:1.1–1.2:1.27, 2015. doi:10.1145/2699875.

17 Karl Däubel, Sven Jäger, Torsten Mütze, and Manfred Scheucher. On orthogonal symmetric chain decompositions. *Electron. J. Comb.*, 26(3):3, 2019. doi:10.37236/8531.

18 Giuseppe Di Battista and Walter Didimo. Gdtoolkit. In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization*, pages 571–597. Chapman and Hall/CRC, 2013.

19 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.

20 Giuseppe Di Battista, Ashim Garg, Giuseppe Liotta, Roberto Tamassia, Emanuele Tassinari, and Francesco Vargiu. An experimental comparison of four graph drawing algorithms. *Comput. Geom.*, 7:303–325, 1997. doi:10.1016/S0925-7721(96)00005-3.

21 Peter Eades, Seok-Hee Hong, and Sheung-Hung Poon. On rectilinear drawing of graphs. In David Eppstein and Emden R. Gansner, editors, *Graph Drawing, GD 2009*, volume 5849 of *LNCS*, pages 232–243. Springer, 2009. doi:10.1007/978-3-642-11805-0_23.

22 Markus Eiglsperger. *Automatic layout of UML class diagrams: a topology-shape-metrics approach*. PhD thesis, University of Tübingen, Germany, 2003. URL: http://w210.ub.uni-tuebingen.de/dbt/volltexte/2004/1028/index.html.

23 Markus Eiglsperger, Ulrich Fößmeier, and Michael Kaufmann. Orthogonal graph drawing with constraints. In David B. Shmoys, editor, *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA*, pages 3–11. ACM/SIAM, 2000. URL: http://dl.acm.org/citation.cfm?id=338219.338225.

24 Ulrich Fößmeier and Michael Kaufmann. Drawing high degree graphs with low bend numbers. In Franz-Josef Brandenburg, editor, *Graph Drawing, Symposium on Graph Drawing, GD '95*, volume 1027 of *LNCS*, pages 254–266. Springer, 1995. doi:10.1007/BFB0021809.

25 Mohammad Ghoniem, Jean-Daniel Fekete, and Philippe Castagliola. On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. *Inf. Vis.*, 4(2):114–135, 2005. doi:10.1057/PALGRAVE.IVS.9500092.

26 Tim Hegemann and Alexander Wolff. A simple pipeline for orthogonal graph drawing. In Michael A. Bekos and Markus Chimani, editors, *Graph Drawing and Network Visualization - GD 2023*, volume 14466 of *LNCS*, pages 170–186. Springer, 2023. doi:10.1007/978-3-031-49275-4_12.

27 Weidong Huang, Seok-Hee Hong, and Peter Eades. Effects of crossing angles. In *2008 IEEE Pacific Visualization Symposium*, pages 41–46, 2008. doi:10.1109/PACIFICVIS.2008.4475457.

28 Steve Kieffer, Tim Dwyer, Kim Marriott, and Michael Wybrow. Hola: Human-like orthogonal network layout. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):349–358, 2016. doi:10.1109/TVCG.2015.2467451.

29 Gunnar W. Klau and Petra Mutzel. Optimal compaction of orthogonal grid drawings. In Gérard Cornuéjols, Rainer E. Burkard, and Gerhard J. Woeginger, editors, *Integer Programming and Combinatorial Optimization, IPCO '99*, volume 1610 of *LNCS*, pages 304–319. Springer, 1999. doi:10.1007/3-540-48777-8_23.

30 Ján Manuch, Murray Patterson, Sheung-Hung Poon, and Chris Thachuk. Complexity of finding non-planar rectilinear drawings of graphs. In Ulrik Brandes and Sabine Cornelsen, editors, *Graph Drawing - GD 2010*, volume 6502 of *LNCS*, pages 305–316. Springer, 2010. doi:10.1007/978-3-642-18469-7_28.

31 Torsten Mütze and Manfred Scheucher. On l-shaped point set embeddings of trees: first non-embeddable examples. *J. Graph Algorithms Appl.*, 24(3):343–369, 2020. doi:10.7155/JGAA.00537.

32 Takao Nishizeki and Md. Saidur Rahman. *Planar Graph Drawing*, volume 12 of *Lecture Notes Series on Computing*. World Scientific, 2004. doi:10.1142/5648.

33 Achilleas Papakostas and Ioannis G. Tollis. Improved algorithms and bounds for orthogonal drawings. In Roberto Tamassia and Ioannis G. Tollis, editors, *Graph Drawing, DIMACS International Workshop, GD '94*, volume 894 of *LNCS*, pages 40–51. Springer, 1994. doi:10.1007/3-540-58950-3_355.

34 Achilleas Papakostas and Ioannis G. Tollis. Interactive orthogonal graph drawing. *IEEE Trans. Computers*, 47(11):1297–1309, 1998. doi:10.1109/12.736444.

35 Maurizio Patrignani. On the complexity of orthogonal compaction. *Computational Geometry: Theory and Applications*, 19(1):47–67, 2001. doi:10.1016/S0925-7721(01)00010-4.

36 Mauricio G.C. Resende and Celso C. Ribeiro. A grasp for graph planarization. *Networks: An International Journal*, 29(3):173–189, 1997. doi:10.1002/(SICI)1097-0037(199705)29:3\%3C173::AID-NET5\%3E3.0.CO;2-E.

37 Manfred Scheucher. Two disjoint 5-holes in point sets. *Comput. Geom.*, 91:101670, 2020. doi:10.1016/J.COMGEO.2020.101670.

38 Manfred Scheucher, Hendrik Schrezenmaier, and Raphael Steiner. A note on universal point sets for planar graphs. *CoRR*, abs/1811.06482, 2018. arXiv:1811.06482.

39 R. Tamassia and I.G. Tollis. Planar grid embedding in linear time. *IEEE Transactions on Circuits and Systems*, 36(9):1230–1234, 1989. doi:10.1109/31.34669.

40 Roberto Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.*, 16(3):421–444, 1987. doi:10.1137/0216030.

41 Roberto Tamassia. Constraints in graph drawing algorithms. *Constraints An Int. J.*, 3(1):87–120, 1998. doi:10.1023/A:1009760732249.

42 Roberto Tamassia, Carlo Batini, and Maurizio Talamo. An algorithm for automatic layout of entity-relationship diagrams. In Carl G. Davis, Sushil Jajodia, Peter A. Ng, and Raymond T. Yeh, editors, *Int. Conf. on Entity-Relationship Approach (ER'83)*, pages 421–439. North-Holland, 1983.

43 Roberto Tamassia, Giuseppe Di Battista, and Carlo Batini. Automatic graph drawing and readability of diagrams. *IEEE Trans. Syst. Man Cybern.*, 18(1):61–79, 1988. doi:10.1109/21.87055.

44 Gopalakrishnan Vijayan and Avi Wigderson. Rectilinear graphs and their embeddings. *SIAM Journal on Computing*, 14(2):355–372, 1985. doi:10.1137/0214027.

Reconfiguration in Curve Arrangements to Reduce Self-Intersections and Popular Faces

Florestan Brunck
University of Copenhagen, Denmark

Hsien-Chih Chang
Dartmouth College, Hanover, NH, USA

Maarten Löffler
Department of Information and Computing Sciences, Utrecht University, The Netherlands
Department of Computer Science, Tulane University, New Orleans, LA, USA

Tim Ophelders
Department of Information and Computing Sciences, Utrecht University, The Netherlands
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Lena Schlipf
AI Center, Department of Computer Science, University of Tübingen, Germany

Abstract

We study reconfiguration in curve arrangements, where a subset of the crossings are marked as *switches* which have three possible states, and the goal is to set the switches such that the resulting curve arrangement has few self-intersections, or few faces that are incident to the same curve multiple times (a.k.a. *popular faces*). Our results are that these problems are NP-hard, but FPT in the number of switches. Minimizing self-intersections is also FPT in the number of *non-switchable* crossings; for minimizing popular faces this problem remains open. Our results can be applied to generating *curved nonograms*, a type of logic puzzle that has received some attention lately. Specifically, our results make it possible to efficiently convert *expert* puzzles into *advanced* puzzles (or determine that this is impossible).

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Fixed parameter tractability

Keywords and phrases Curve Arrangements, Reconfiguration, Curve Arrangements, NP-hardness, Fixed-Parameter Tractability, Puzzle Generation

Digital Object Identifier 10.4230/LIPIcs.GD.2025.36

Funding *Tim Ophelders*: Partially supported by the Dutch Research Council (NWO) under project no. VI.Veni.212.260.

1 Introduction

We study reconfiguration problems in curve arrangements in the plane. A set of open or closed curves that are well-behaved induces an *arrangement* that partitions the plane into faces, edges, and vertices; we consider a *switch* operation that locally uncrosses two crossing curves [7, 8, 10, 9, 14, 11]. Note that under this operation each arrangement vertex has three possible "states", but in two of the three states the vertex will no longer be a vertex after the switch; we therefore define *switches* to be local regions which either contain a vertex (crossing) or two disjoint strands of curves, see Figure 1.

Our terminology is formally defined in Section 1.2. We are interested in reducing the number of self-intersections and *popular faces* [5] in the arrangement using – ideally few – such switch operations.


© Florestan Brunck, Hsien-Chih Chang, Maarten Löffler, Tim Ophelders, and Lena Schlipf;
licensed under Creative Commons License CC-BY 4.0

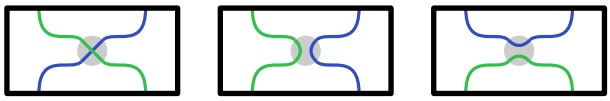


Figure 1 The three states of a switch. Only the left one contains a crossing and hence a vertex.

1.1 Motivation & Background

Our question is motivated by the problem of generating *curved nonograms*. Nonograms, also known as *Japanese puzzles*, *paint-by-numbers*, or *griddlers*, are a popular puzzle type where one is given an empty grid and a set of *clues* on which grid cells need to be colored. A clue consists of a sequence of numbers specifying the numbers of consecutive filled cells in a row or column. A solved nonogram typically results in a picture (see Figure 2(a)). There is quite some work in the literature on the difficulty of solving nonograms [1, 3, 15, 17].

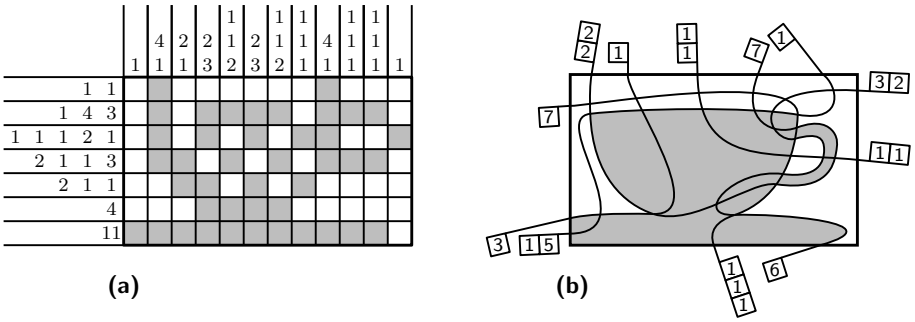


Figure 2 Two nonogram puzzles in solved states. **(a)** A classic nonogram. **(b)** A curved nonogram. (Puzzles in Figures 2 and 3 are part of the supplementary material of van de Kerkhof et al [16].)

Van de Kerkhof et al. [16] introduced *curved* nonograms, a variant in which the puzzle is no longer played on a grid but on any arrangement of curves (see Figure 2(b)). In curved nonograms, a clue specifies the numbers of filled faces of the arrangement in the sequence of faces that are incident to a common curve on one side. Van de Kerkhof et al. focused on heuristics to automatically generate such puzzles from a desired solution picture by extending curve segments to a complete curve arrangement. They observed that curved nonograms come in different flavours of increasing complexity – not in terms of how hard it is to *solve* a puzzle, but how hard it is to understand the rules (see Figure 3).

- *Basic* nonograms are puzzles in which each clue corresponds to a sequence of unique faces. The analogy with clues in classical nonograms is straightforward.
- *Advanced* nonograms may have clues that correspond to a sequence of faces in which some faces appear more than once because the face is incident to the same curve multiple times. When such a face is filled, it is also counted multiple times; in particular, it is no longer true that the sum of the numbers in a clue is equal to the total number of filled faces incident to the curve. This makes the rules harder to understand.
- *Expert* nonograms may have clues in which a single face is incident to the same curve on *both* sides. They are even more confusing than advanced nonograms.

Löffler et al. [12] strenghten this classification by observing that the different flavours of nonograms also behave differently in terms of complexity of solving - in particular, they show that the existing concept of *simple* classical nonograms [1] can be extended to curved nonograms, and that *basic* and *advanced* simple curved nonogram can still be solved in polynomial time while solving *expert* simple curved nonograms is likely to be hard.

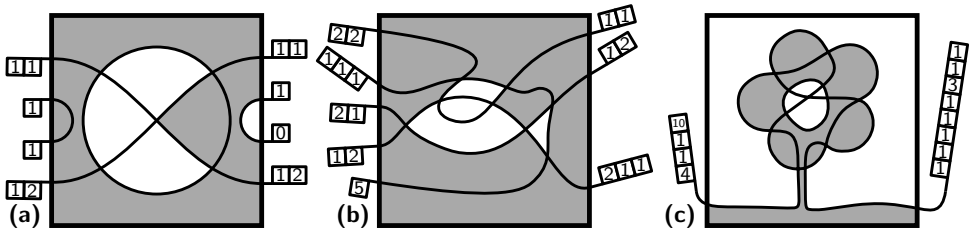


Figure 3 Three types of curved nonograms of increasing complexity [16], shown with solutions. **(a)** *Basic* puzzles have no popular faces. **(b)** *Advanced* puzzles may have popular faces, but no self-intersections. **(c)** *Expert* puzzles have self-intersecting curves. We can observe closed curves (without clues) in **(a)** and **(c)**.

Van de Kerkhof et al. stated that it would be of interest to generate puzzles of a specific complexity level. Their generators are currently not able to do so other than by trial and error. Specifically, the puzzles they generate are almost always of the "expert" complexity and occasionally of the "advanced" complexity; particularly the probability of generating large basic puzzles is so small that trial and error is not a feasible approach.

De Nooijer et al. [4, 5, 6] explore one possible way to generate nonograms of a specific complexity: rather than building a new generator from scratch, they suggest using an existing generator and modifying the output. They define a *popular face* in a curve arrangement to be a face that is incident to a single curve multiple times. With this definition, expert puzzles correspond exactly to arrangements with self-intersecting curves, and advanced puzzles correspond exactly to the presence of *popular faces* in the arrangement. Note that a self-intersection necessarily implies at least one popular face.

De Nooijer et al. investigated how to reduce the number of popular faces by *inserting* new curves into the arrangement. This approach ensures that the shapes of the faces are not distorted. But the approach also has some drawbacks: by definition it increases the size of the arrangement; also, by inserting curves we will never remove any self-intersection, so this approach is only suitable for transforming advanced puzzles into basic puzzles but not expert puzzles into advanced ones.

In this paper, we explore a different approach based on local changes to the arrangement. This will keep the size of the arrangement and its visual complexity roughly the same. However, in order to reconnect curves locally it is necessary to distort the curves a little bit, which may be undesirable for curve pieces that outline vital parts of the solution picture. Therefore, we assume that a set of *switches* is given in advance: (arbitrarily) small regions in which the arrangement may be reconfigured. Such a set of switches could be obtained either by human intervention, or automatically (for instance, one could take all crossings in the arrangement that are sufficiently far from the solution picture).

1.2 Terminology & Preliminaries

Let $\mathcal{A}$ be a set of curves which lie inside the area bounded by a closed curve F, called the *frame*. All curves in $\mathcal{A}$ are either *closed* or they are *open* with a start and end point on F. We refer to $\mathcal{A}$ as a *curve arrangement*, see Figure 4(a). We consider only *simple* arrangements, where no three curves meet in a point, and all intersections are transversal crossings (no tangencies). The arrangement $\mathcal{A}$ can be seen as an embedded multigraph whose vertices are crossings between curves and whose edges are *curve segments*. Arrangement $\mathcal{A}$ subdivides the region bounded by F into *faces*. We call a face *popular* when it is incident to multiple curve segments belonging to the same curve in $\mathcal{A}$ (see Figures 4(b–c)).

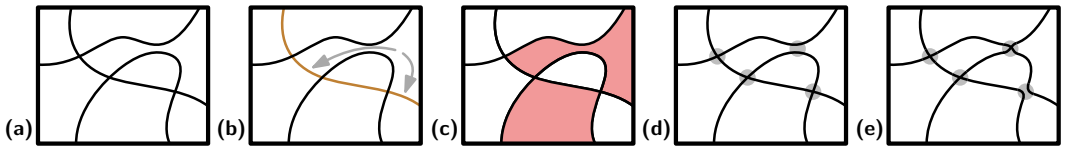


Figure 4 **(a)** An arrangement of curves inside a frame. **(b)** The highlighted curve is incident to the top right face in two segments, making the face *popular*. **(c)** All popular faces are highlighted. **(d)** A set of switches. **(e)** A possible reconfiguration after which no more faces are popular.

Now, let a *switch* be a tiny topological disk around each crossing, which is traversed by exactly two curve segments, in which we are allowed to reroute the curves of $\mathcal{A}$ (see Figures 4(d–e)). We study the problem:

▶ **Problem 1.** *Given a curve arrangement and a set of switches, can we reconfigure the curves so as to remove all, or as many as possible, self-intersections and/or popular faces? And can we minimize the number of switch operations?*

Once we are given a set of switches, the question above is essentially combinatorial. Observe that the problem is only interesting if the arrangement has at least some *non-switchable* crossings (or we want to minimize the number of switch operations).

▶ **Observation 2.** *If all intersections are switches, it is possible to remove all popular faces.*

Proof. Simply set every switch to any non-intersecting state. Then each curve bounds a single face on each side, and thus no face is popular. ◀

1.3 Results & Organization

We make the following contributions. In Section 2, we show that the problem of testing whether it is possible to remove either all self-intersections or all popular faces from an arrangement, where some vertices are marked as switchable and some as non-switchable, is NP-hard. On the other hand, when all vertices are switchable, the problem is trivial (Observation 2). However, we do show that in this case finding a solution that minimizes the number of switched vertices is also NP-hard. In Section 3, we turn our attention to parameterized results. We show that when we have an arrangement with only k switchable crossings, or k *non-switchable* crossings, the problem of removing all self-intersections is fixed-parameter tractable in k. For k switchable crossings, the same is true for removing the popular faces. Finally, we show that another natural parameter, the number of popular faces in the input, does not help: the problem remains NP-hard even when the input has a constant number of popular faces.

2 NP-hardness

In this section, we present NP-hardness results and bring a negative answer to Problem 1. We begin by examining the problem of removing all self-intersections and derive Theorem 3, using an intermediate *Permuter problem*. From there, we then reduce the problem of removing popular faces using a technique of overlaying a *global* grid. Finally, we present subsequent extensions concerning the minimal number of switch operations required by overlying *local* grids, or *waffles*. The organisation of our results in this section is summarized in Figure 5.

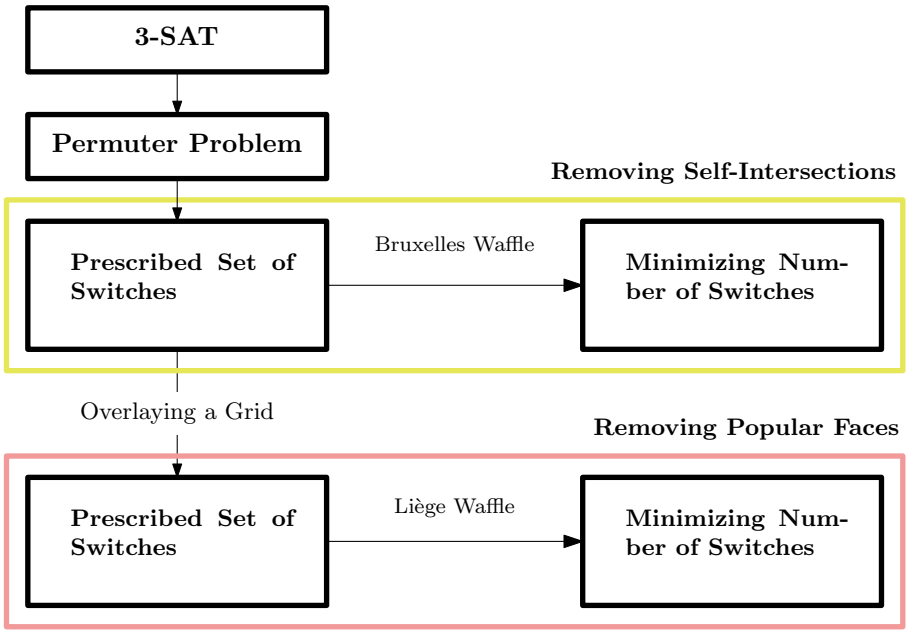


Figure 5 The flow of successive reductions underlying our NP-hardness proofs. Each arrow represents a reduction.

2.1 Base Results

We start by proving our base results: we introduce the *Permuter problem*, we show that 3-SAT reduces to the Permuter problem, and we show that the Permuter problem reduces to removing self-intersections in curve arrangements.

▶ **Theorem 3.** *Given a curve arrangement and a prescribed set of switches, it is NP-hard to decide whether it is possible to configure the switches such that the resulting arrangement has no self-intersections.*

2.1.1 The Permuter Problem and its Reduction from 3-SAT

Consider the straight-line drawing of the complete bipartite graph $K_{k,k}$ in the plane, with bi-partition I and O (referred as its *inputs* and *outputs*) such that I and O are evenly distributed on opposite sides of a rectangle R. Fixing the same linear horizontal order on the vertices of I and O, the *k-permuter* $\Pi_{k,\sigma}$ is the matching of $K_{k,k}$ associated to the permutation σ of $[n]$. An instance of the Permuter problem consists of a finite collection $\{\Pi_{k_i}, \sigma_i\}_{i\in[n]}$ of k_i-permuters, realised in the plane using an associated collection $\{R_i\}_{i\in[n]}$ of rectangles, together with a finite collection of paths $\{P_i\}_{i\in[m]}$ outside the rectangles, such that:

- Every element of $\{P_i\}_{i\in[m]}$ has both its endpoints in one of the k-permuters (possibly the same, and possibly both inputs/outputs).
- Every input and output of every permuter is connected to a unique element of $\{P_i\}_{i\in[m]}$.
- For all $i \in [m]$, for all $j \in [n]$, $P_i \cap R_j = \varnothing$; i.e., each path P_i is outside each rectangle R_j.

By construction, every path of $\{P_i\}_{i\in[m]}$ belongs to a unique closed loop. The *Permuter problem* then consists in deciding whether or not there exists a choice of n permutations $\sigma_1, \sigma_2, \ldots, \sigma_n$ of $[k_1], [k_2], \ldots, [k_n]$ such that each resulting loop is *simple* (see Fig. 6).

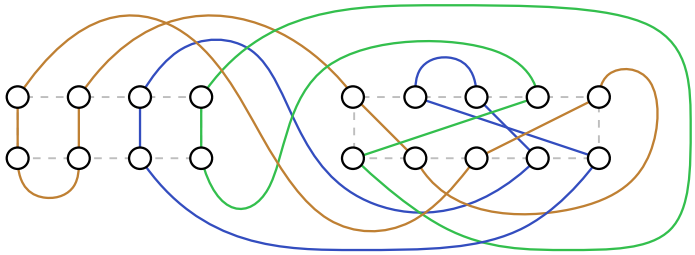


Figure 6 An instance of the k-Permuter problem with a 4-permuter and a 5-permuter, assigned with the permutations $\sigma_1 = \mathrm{Id}$ and $\sigma_2 = (12534)$. There are 3 resulting closed loops: one simple (green) and two self-intersecting (blue and brown).

We believe that the Permuter problem is of independent interest, and we show that it is NP-hard. We will prove Theorem 3 by reduction from the Permuter problem.

▶ **Theorem 4.** *The Permuter problem is NP-hard.*

Proof. The proof is by a polynomial-time reduction from 3-SAT. We shall use two types of gadgets: 2-permuters for variable assignment and 3-permuters for clause verification (see Fig. 7). For simplicity, each k-permuter is depicted by a black box on the diagram, where the value of k is made clear by the number of incoming/outgoing paths. Each different colour in the figure indicates a different variable. The thick or thin dashed lines on the top, bottom and middle-left part of the diagram indicate respectively the false and true literals of each variable. The thick and thin solid lines in the middle-right section of the diagram indicate respectively the true or false assignment of each variable. Given a Boolean formula in 3-CNF form with n variables, we construct $2n$ non-crossing semi-circular arcs. We replicate this construction twice to form the top and bottom parts of the diagram. In the middle, we show a single clause gadget, involving two 3-permuters. To simulate the two logical OR of the clause, we proceed as follows: if the corresponding 3-clause involves the variables x_i, x_j and x_k, we select the wire *opposite* to their desired truth value in the clause (i.e. thick for $\overline{x_i}$, thin for x_i) and "drag" them towards the gadget to intersect the same single path chosen among the three paths linking the output of the first 3-permuter to the inputs of the second. By construction, there is no valid permutation assignment to the two 3-permuters which avoids all possible self-intersections with the three black paths if and only if x_i, x_j and x_k all have the wrong truth assignment. Furthermore, to the right of the 3-clause gadget, we have weaved the incoming paths of the first and the outgoing paths of the second in such a way that, if the composition of the two 3-permuters were not the identity, at least one of the resulting closed loop would self-intersect. Thus each such pair of 3-permuters cannot "cheat" and has to compose to the identity. As a consequence, for each of the variables involved, the composition of its two 2-permuters must also be the identity. By construction, there are then exactly two ways of ensuring this is the case: either both 2-permuters are the identity themselves (setting the variable to be true), or both of them correspond to the transposition (12) swapping the inputs (setting the variable to be false). Fig. 8 shows the instance created for the Boolean formula $(\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_4} \vee x_5 \vee \overline{x_6})$. ◀

2.1.2 The Permuter Problem Reduces to the Self-Intersection Problem

▶ **Lemma 5.** *The problem of configuring a given set of switches to avoid self-intersections is polynomial-time reducible from the Permuter problem.*

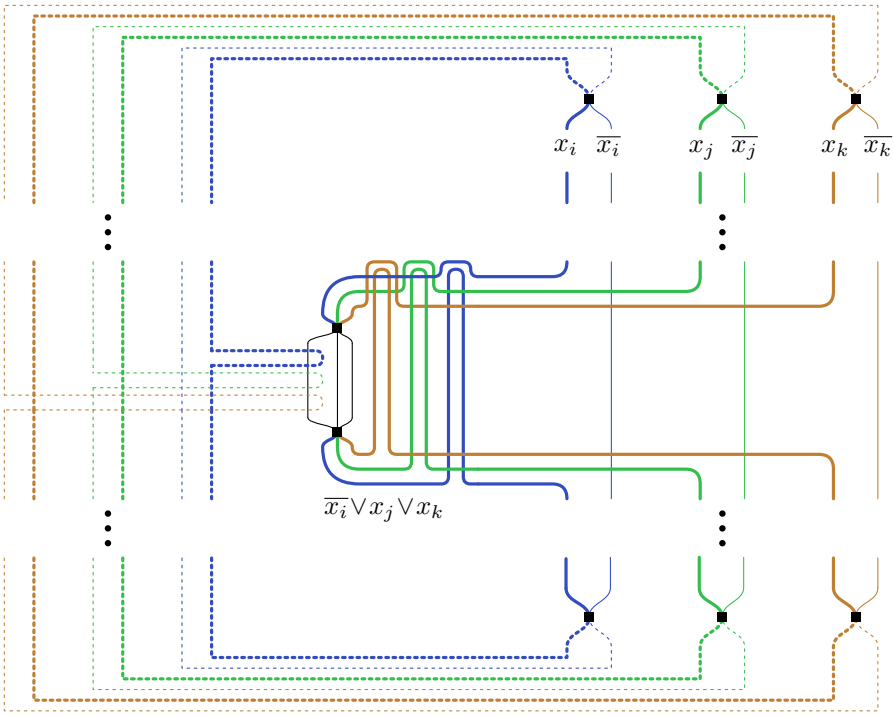


Figure 7 A single 3-clause in the reduction from 3-SAT to the Permuter problem.

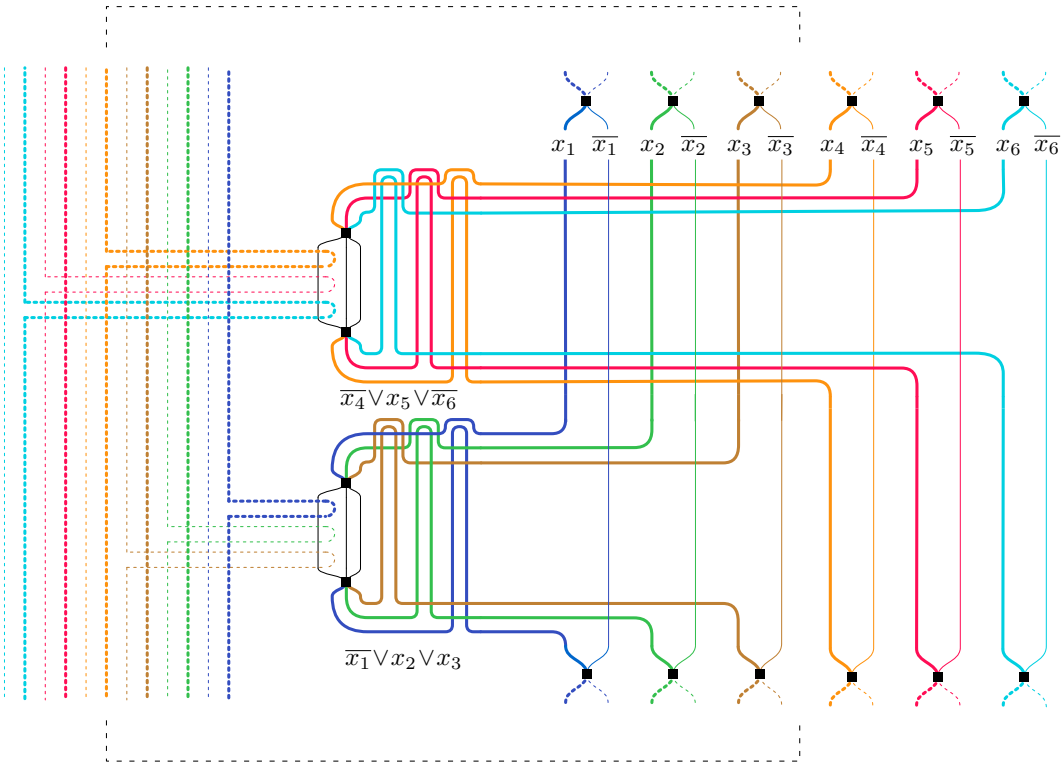


Figure 8 The instance of the Permuter problem created for the Boolean formula $(\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_4} \vee x_5 \vee \overline{x_6})$.

Proof. The proof of the claim proceeds as follows: we first construct self-intersection gadgets to simulate 2-permuters, take note of the fact that S_n is generated by transpositions and show how to use 2-permuters to construct general k-permuters. The construction for 2-permuters is presented on Fig. 9: the inputs and outputs are connected by two curves weaved into a double coil structure with two intersections, one of which is a switch. The three resulting configurations are shown on the figure; only the leftmost two are free of self-intersections.

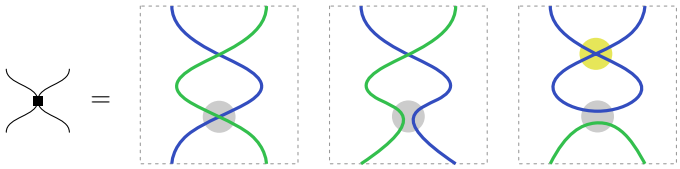


Figure 9 Constructing a 2-permuter using two "doubly-coiled" curves. The grey disk indicates the only switch available, the yellow disk highlights the self-intersection forbidding the third possibility given by the switch.

To simulate a general k-permuter, we introduce the gadget illustrated in Fig. 10. We begin with a k-by-k square and evenly distribute k inputs and outputs on its top and bottom edges, respectively. For every $i \in [n]$, the top i-th input is connected to the bottom $(n-i)$-th output by the path of slope -1 which gets reflected into a path of slope 1 upon meeting the left edge of the square. We then insert a total of $\frac{k(k-1)}{2}$ 2-permuters: one at every site where two paths intersect inside the square.

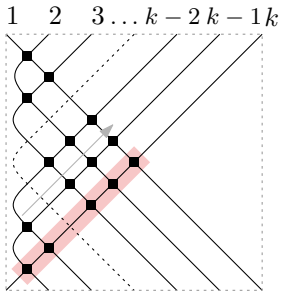


Figure 10 Constructing a k-permuter, using $\frac{k(k-1)}{2}$ 2-permuters.

While many configurations of this gadget are redundant and yield the same permutation, a short inductive argument shows that it is indeed able to simulate any permutation on k elements. The base case is simply the 2-permuter we previously described. Assume then by induction that the version of our gadget with $k-1$ inputs and outputs can successfully simulate all permutations of $[k-1]$. Note that any permutation σ of $[k]$ can be written as the composition of a permutation of $[k-1]$ followed by the insertion of the element k into one of the k positions before or after one of the $k-1$ permuted elements. It is thus enough to show that the addition of the last "row" of $k-1$ 2-permuters (highlighted in pink) can simulate this last insertion step. Labelling the $(k-1)$ 2-permuters of the $(k-1)$-th row according to the direction indicated on Fig. 10, we insert the element k in position i (before the i-th element) by setting all the 2-permuters from positions i to k to swap their inputs, and the remaining i 2-permuters to the identity. This effectively shifts all the elements with positions greater than i by 1 (the permuters reroute their corresponding wires to the segment of slope 1 instead of -1) as we sequentially shift the element k to the left i times (the permuters successively let the k-th input path "slide" on the last path of slope -1). ◀

Proof of Theorem 3. Theorem 3 now follows directly from Theorem 4 and Lemma 5. ◀

2.2 Extensions

We now extend the result from Theorem 3 in several ways, to prove that the problem remains hard when we wish to minimize the number of switch operations, or when the goal is to remove all popular faces rather than self-intersections. The idea is always to locally alter the reduction in a way that does not affect its global properties.

2.2.1 Minimizing the number of switches

In the previous section, we had a prescribed set of switches; indeed, this is necessary since if we are allowed to switch everywhere, then we can always remove all self-intersections by Observation 2.

However, now consider the scenario in which we wish to minimize the number of switch operations, or, in the decision version, we wish to test for a given k whether there exists a sequence of k switch operations such that the resulting arrangement has no self-intersections. In this scenario, we may or may not have a prescribed set of switches.

The idea is to emulate the construction from Section 2.1, but to replace every self-intersection in the construction which is *not* a switch by a *waffle gadget*. Such a gadget is built in such a way that even if *every* intersection in the gadget is a switch, the number of switches required to change its global state is more than a parameter c. If we then choose $c > k$, the result follows since essentially we are never allowed to switch these gadgets.

The Bruxelles Waffle. We introduce the *Bruxelles waffle* (in contrast to the *Liège waffle* which we describe in Section 2.2.3). The construction is illustrated in Figure 11.

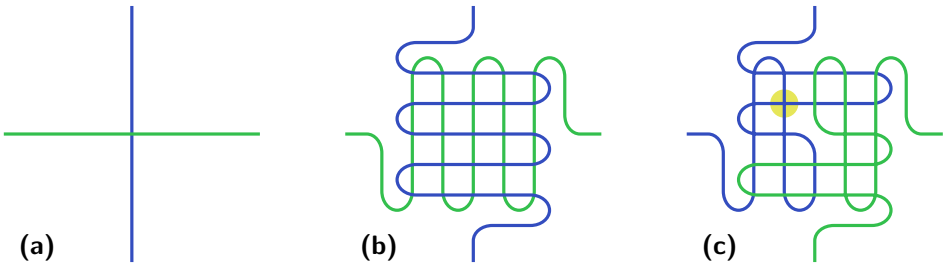


Figure 11 The Bruxelles waffle. (a) A crossing between two different curves. (b) To construct a Bruxelles waffle at the crossing, we adapt the two curves so they intersect c^2 times. (c) Any sequence of local switches to emulate a global switch must be of length at least c or otherwise lead to a self-intersection.

▶ **Lemma 6.** *In a Bruxelles waffle, any sequence of fewer than c switches must result in an arrangement with either the same combinatorial structure as the original, or at least one self-intersection.*

Proof. Assume that after fewer than c switches, no self-intersections remain. We need to show that opposite terminals lie on the same strand. Assume for a contradiction that opposite terminals do not lie on the same strand. Then the left terminal lies on the same strand as either the top or the bottom terminal. Without loss of generality (by rotational symmetry of the gadget) assume that the left terminal lies on the same strand as the bottom terminal.

Because there are c rows, at least one row has not undergone any switch, and the horizontal path in that row has a single color. Because there are no self-intersections, all strands crossing the row vertically have a color different from the horizontal path of the row.

Removing the row splits the gadget into a part above and a part below the row, and the left and right endpoints of the horizontal path connect to different such parts. The strand containing the horizontal path of the row cannot be closed up without crossing the row, so the strand connects a terminal below and a terminal above the row. That is, it connects the bottom or left terminal to the top or right terminal. This contradicts our assumption that the bottom and left terminals lie on the same strand. ◀

▶ **Theorem 7.** *Given a curve arrangement and an integer k, it is NP-hard to decide whether there exists a sequence of k switches such that the resulting arrangement has no self-intersections.*

2.2.2 Removing popular faces

Next, we consider the problem of removing all popular faces rather than only self-intersections.

The idea is to globally overlay the construction from Section 2.1 with a sufficiently fine grid of horizontal and vertical lines, in which none of the intersections with these new lines are switches. In particular, we make the grid such that each cell of the grid contains either a single strand of one of the curves, or two strands that intersect, or nothing at all (see Figure 12).

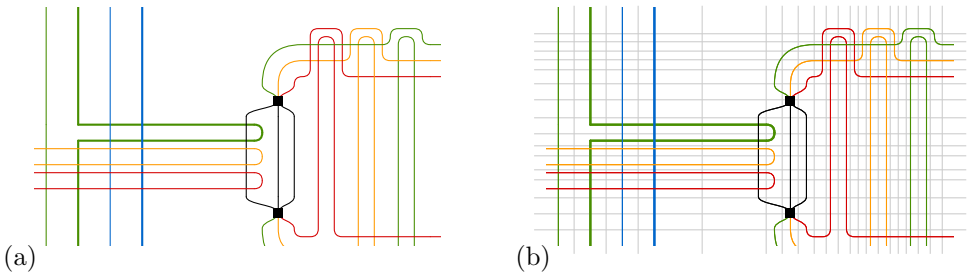


Figure 12 (a) A small section of the construction. (b) A fine grid that separates all elements of the construction into separate cells.

After we overlay the grid, all (new) faces are incident to each grid line only once, and incident to each original curve only once, unless two crossing strands inside a grid cell belong to the same curve. In other words, all remaining popular faces are now due to self-intersections. Therefore, in order to remove all popular faces we need to remove all self-intersections, and this is also sufficient.

▶ **Theorem 8.** *Given a curve arrangement and a prescribed set of switches, it is NP-hard to decide whether it is possible to configure the switches such that the resulting arrangement has no popular faces.*

2.2.3 Removing popular faces with a minimum number of switches

Finally, we consider the setting where every intersection is an allowed switch; in this case, testing whether all popular faces can be removed is again not hard by Observation 2. However, when we wish to remove all popular faces using a minimum number of switch operations, the problem remains NP-hard. The idea is similar to that in Section 2.2.1, but we will need to use a different gadget that ensures we cannot perform any switches (since having no self-intersections does not necessarily imply there are no popular faces).

The Liège Waffle. We introduce the *Liège waffle*, illustrated in Figure 13.

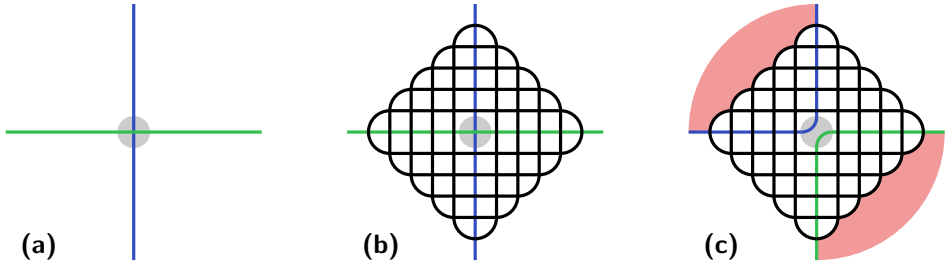


Figure 13 The Liège waffle. (a) A crossing between two different curves. (b) To construct a Liège waffle, we insert a set of c new closed curves that surround the crossing. (c) Any sequence of local switches to emulate a global switch must be of length at least c or lead to a popular face.

Essentially, we overlay c new closed curves on the two (crossing) terminal strands, such that each of the new curves is incident to each of the four unbounded faces, and that the disk bounded by the curve contains the crossing between the two terminal strands.

▶ **Lemma 9.** *In a Liège waffle, any sequence of fewer than c switches must result in an arrangement with either the same combinatorial structure as the original, or at least one popular face.*

Proof. If we change the global connectivity, then one of the four unbounded faces will globally have two strands of the same curve; say the top left face (Figure 13 (c)). In order for this face to *not* be popular, these two strands must be consecutive along the curve. Initially, all intersections incident to the face are crossing, so they must all be uncrossed. There are $c+1$ such intersections, so we need at least $c+1$ switches. ◀

▶ **Theorem 10.** *Given a curve arrangement and an integer k, it is NP-hard to decide whether there exists a sequence of k switches such that the resulting arrangement has no popular faces.*

3 Parameterized Results

Since the problem in general is NP-hard, we now turn our attention to parameterised complexity. We can identify several natural parameters:

- the number of popular faces in a provided configuration; or
- the number of switches; or
- the number of non-switchable crossings.

The number of popular faces is perhaps the most natural parameter when trying to remove popular faces; in particular, existing heuristics for generating curve arrangements do attempt to minimize the number of popular faces and often succeed in reducing their number, but usually not all [16]. Unfortunately, as we show in Section 3.1, this does not make the problem easier. Instead, we can also consider the number of switches as a parameter. When there is only a small number of switches, it is easy to see that the problem becomes polynomial, see Section 3.2. However, in our nonogram generation application, it seems more reasonable to assume the number of *non-switchable* crossings is small: we may wish to maintain some key features of the solution image, but otherwise want to have as much freedom as possible. Here, we prove the problem is also in FPT when the goal is to reduce the number of self-intersections; for removing popular faces, the problem remains open. In Section 3.3, we explore this parameter in detail. Our parameterized results are summarized in Table 1.

Table 1 Results on parameterized complexity.

parameter	avoid self-intersections	avoid popular faces
number of popular faces	NP-hard	NP-hard
number of switchable crossings	FPT	FPT
number of non-switchable crossings	FPT	open

3.1 Parameter: number of faces

We show that the problem of removing all popular faces remains hard if we are given a starting configuration with only a constant number of popular faces. For this, we set the permuters in the construction in Section 2.1.2 to some an initial state that leads to only few popular faces but does not "help" with finding the actual solution. In particular, an assignment of a 3-SAT formula that satisfies all but one clause corresponds to a setting of permuters that results in constantly many popular faces.

We first show that 3-SAT remains hard if the input comes with an assignment that violates at most one clause; call this problem Almost-3-SAT.

▶ **Lemma 11.** *Almost-3-SAT is NP-complete.*

Proof. NP-membership is trivial. To show NP-hardness, assume without loss of generality that P≠NP, and that we have a polynomial time algorithm Alg for deciding Almost-3-SAT. We can use Alg to also output a satisfying assignment if the instance is satisfiable (by guessing the value of a variable, removing all clauses satisfied by that variable, and calling Alg on the resulting formula). We hence assume that Alg also outputs a satisfying assignment (if one exists).

We show that we can use Alg to solve 3-SAT. Consider an instance F of 3-SAT that consists of m clauses. We can solve F by invoking Alg m times, namely on subformulas of F. Suppose that we have an assignment A that satisfies the first k clauses of F. Then the first $k+1$ clauses of F together with A form an instance of Almost-3-SAT, which we can solve using Alg to either obtain an assignment A' that satisfies the first $k+1$ clauses of F, or a proof that the first $k+1$ clauses of F are not satisfiable, and hence F is not satisfiable. Starting with $k=0$, and repeating the above m times and incrementing k, we either obtain a satisfying assignment for F, or discover that F is not satisfiable. Hence, Alg can be used to solve 3-SAT in polynomial time. ◀

▶ **Theorem 12.** *Given a curve arrangement with a constant number of popular faces, the problem of deciding whether there exists a configuration without any popular faces (or self-intersections) is still NP-complete.*

Proof. Setting the permuters to an assignment that satisfies all except one clause results in configuration in which all popular faces are incident to the clause gadget that corresponds to the unsatisfied clause. Because a clause gadget has constant complexity, the configuration has a constant number of popular faces. The popular faces can be removed by reconfiguring switches if and only if the instance of Almost-3-SAT is satisfiable. ◀

3.2 Parameter: number of switches

Next, we show that the problem is not NP-hard when the number of switches is small.

▶ **Observation 13.** *Given a curve arrangement with k switches, the problem of deciding whether there exists a configuration without any popular faces (or self-intersections) is FPT in k.*

Proof. We can simply try all 3^k combinations and then check whether the resulting arrangement has self-intersections and/or popular faces. Both of these can be done in time proportional to the complexity of the arrangement. So, the running time of this algorithm is $O(3^k \cdot n)$. ◀

3.3 Parameter: number of non-switchable crossings

Finally, we explore our third parameter: the number of *non-switchable* crossings in the input arrangement. Throughout this section, we will denote this parameter as k. The case $k = 0$ corresponds to Observation 2 and is easy. We will first explore the situation when $k = 1$.

3.3.1 Removing self-intersections with exactly 1 non-switchable crossing

This question is already interesting when there is just *one* crossing that cannot be switched. As a simple example, consider the arrangement in Figure 14, which shows an arrangement with five switches, one of which is marked as non-switchable. There are exactly three states in which there are no self-intersections, but to get from one to another, four switch operations need to be done; that is, every single switchable switch needs to be flipped.

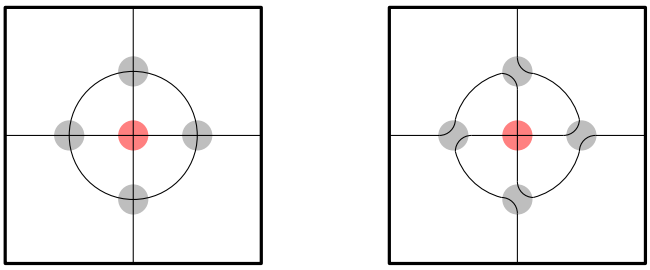


Figure 14 An arrangement with five switches, the middle of which is non-switchable. Left: one possible state without self-intersections. Right: another state without self-intersections. A third state is the mirror image of the right figure.

Define an Eulerian partition of a multigraph to be set of cycles and paths connecting boundary edges, such that each edge is used exactly once. That is, each edge lies on a single path or cycle, and no path or cycle revisits an edge. We say that an Eulerian partition is simple if each of its paths and cycles is simple. For a surface-embedded graph, we endow each vertex (of even degree) with the involution that pairs up incident edges on "opposite" sides. An Eulerian partition is straight at a vertex v if whenever a path or cycle passes through v, it does so using the edges related to each other by the involution.

Let G be the graph of the arrangement of the curves, where each switch and non-switch corresponds to a vertex of G, and every maximal segment of the curves becomes an edge of G. Assume that G has a single (degree 4) vertex v that is a non-switchable crossing. We are looking for a simple Eulerian partition of G that is straight at v. Let $G - v$ be the graph obtained by subdividing the four edges incident to v using (in cyclic order) vertices $v_1, \ldots, v_4$, and then removing v. We call v a cut-vertex if $G - v$ has more components than G.

▶ **Lemma 14.** *For any graph G that comes from a curve arrangement, and any non-boundary vertex v of G, exactly one of the following is true:*

1. *There exists a simple Eulerian partition of G that is straight at v; or*
2. *v is a cut-vertex and at least one component of $G - v$ incident to v_i $(i = 1, \ldots, 4)$ does not touch the boundary.*

Proof. We first show that Item 2 rules out Item 1. Let C be a component of $G - v$ that does not touch the boundary and without loss of generality assume that it contains v_1 (otherwise relabel the v_i's). Because C does not touch the boundary, the vertices of C of degree 1 form a subset of $\{v_1, v_2, v_3, v_4\}$. Because v is a cut-vertex of G, there exists some v_i, $i \in \{2, 3, 4\}$ that does not lie in C. By the handshaking lemma, C contains exactly two vertices of degree 1.

We claim that C *does not contain* v_3. Suppose for a contradiction that C contains v_3, then the path from v_1 to v_3 corresponds to a cycle c through v in G that by the Jordan curve theorem separates the components C_2 and C_4 containing v_2 and v_4 respectively, so $C_2 \neq C_4$. By the handshaking lemma, C_2 and C_4 must have an odd (and therefore nonzero) number of boundary ports (to compensate for the vertex v_2 or v_4 of degree 1). However, C_2 or C_4 lies interior to the disk bounded by c and hence has no boundary ports, which is a contradiction, so C does not contain v_3.

Assume without loss of generality that C contains v_1 and v_2 (the argument for v_1 and v_4 is symmetric). Now, if the Eulerian partition of G that was straight at v is simple, then $G - v$ must contain an Eulerian partition that does not contain a path connecting v_1 to v_2 (otherwise the path would self-intersect at v in G). However, because these are the only vertices of C of degree 1, and Eulerian partition of $G - v$ contains such a path, so Item 2 rules out Item 1.

It remains to construct a simple Eulerian partition (that is straight at v) if v is not a cut-vertex, or if all components of $G - v$ incident to v_i ($i \in \{1, \ldots, 4\}$) touch the boundary. Let C_i ($i = 1, \ldots, 4$) be the component of $G - v$ containing v_i. We first show that there exists a simple Eulerian partition that is straight at v if $C_1 = C_3$ (and by symmetric argument if $C_2 = C_4$). Assume that $C_1 = C_3$, then $G - v$ contains a simple path π_1 from v_1 to v_3. Let $G - v - \pi_1$ be the graph obtained from $G - v$ by removing the edges of π_1. If v_2 and v_4 are connected in $G - v - \pi_1$, then there exists a simple path π_2 from v_2 to v_4. Then π_1 and π_2 correspond to simple cycles c_1 and c_2 in G that are straight at v, and the remainder of G admits a Eulerian partition (and hence a simple one). If instead v_2 and v_4 are not connected after removing the edges of π_1 from $G - v$, then we pick an arbitrary Eulerian partition of $G - v - \pi_1$. This partition corresponds to a partition of G in which π_1 corresponds to a simple cycle that is straight at v, and the paths that contain v_2 and v_4 (call them π_2 and π_4) do not intersect because they lie in different components of $G - v - \pi_1$. Hence, the paths π_2 and π_4 can be concatenated by replacing the end points v_2 and v_4 by the vertex v in G, to obtain a simple path in G that is straight at v.

We have established that there exists a simple Eulerian partition that is straight at v if $C_1 = C_3$ or $C_2 = C_4$, so assume that $C_1 \neq C_3$ and $C_2 \neq C_4$. Then v is a cut-vertex, so it remains to show that if each C_i touches the boundary, then there exists a simple Eulerian partition that is straight at v. We claim that $G - v$ contains edge-disjoint simple paths π_i ($i = 1, \ldots, 4$) that connect each v_i to the boundary. For this, let π_1 and π_3 be arbitrary simple paths and consider the component C_2' of $G - v - \pi_1 - \pi_3$ that contains v_2. We claim that this component contains a boundary vertex. Indeed, by the handshaking lemma, C_2' contains an even number of vertices of degree 1, and one of them is v_2. None of the vertices v_1, v_3, v_4 lie in C_2', because v_1 and v_3 were isolated by removing the edges of π_1 and π_3, and v_4 lies on a different component because $C_2 \neq C_4$. Hence, C_2' contains a boundary vertex, and hence a simple path π_2 to it from v_2. By a symmetric argument, a component of $G - v - \pi_1 - \pi_3$ contains a simple path π_4 from v_4 to a boundary vertex. Because $C_1 \neq C_3$ and $C_2 \neq C_4$, the "concatenations" of π_1 with π_3 and of π_2 with π_4 in G are simple and straight at v. The remainder of G (after removing these two paths) admits a Eulerian partition and hence a simple one. Reinserting the two paths yields a simple Eulerian partition of G that is straight at v. ◀

This now leads to the following result:

▶ **Theorem 15.** *For an instance with exactly 1 non-switchable crossing, we can decide whether there exists a configuration without self-intersections in $O(n^2)$ time.*

Proof. Suppose we have an arrangement $\mathcal{A}$ for which the answer is yes, and let G be its (planar) graph. G has one special vertex ξ that represents the non-switchable crossing. We augment G by adding a new vertex ω that represents the outer face; we create an edge from all vertices on the frame to ω Now consider a configuration K for $\mathcal{A}$ without self-intersections. All curves in $\mathcal{A}$ are now edge-disjoint cycles in G. Now, consider the cycles that pass through ξ. Note that these must be two cycles; we will refer to them as the *red* cycle and *blue* cycle.

Now, replace ξ by four degree 1 vertices that we refer to as *terminals*. Let G' be the resulting (planar) graph. Note that now, we have a red and a blue *path* in G', and the paths incident to opposing terminals have the same color.

Now, our goal is to test whether such a configuration K can exist, based only on the information in G. That is, we need to find whether there exist two edge-disjoint paths in G' that connect opposite pairs of terminals. The existence of edge-disjoint paths that connect two pairs of terminals can be tested in $O(n^2)$ time [13].

If so, by Lemma 14, we then have a Eulerian partition of the remainder, and we can cover it by edge-disjoint cycles. In addition, when we re-insert ξ, the two paths become an edge-disjoint red and blue cycle that are straight at ξ. ◀

The proof of Theorem 15 is constructive, and leads to the following algorithm:

▶ **Algorithm 16.**

- *Construct G' by connecting all boundary vertices to an additional vertex ω on the outer face, and replacing ξ by four terminals.*
- *Assign alternating colors to the terminals.*
- *Run [13] (in $O(n^2)$ time).*
- *If it returns false, return false. Otherwise:*
- *For the two paths found by [13], set all switches along the paths accordingly so that two curves will follow exactly these paths.*
- *For the remainder, (greedily) cover G' with edge-disjoint cycles. If any cycle has a self-intersections, simply split it into multiple cycles.*
- *Set all remaining switches so that each cycle corresponds to a curve.*

3.3.2 Removing self-intersections with k non-switchable crossings

Next, we can extend the ideas from the previous section to k non-switchable crossings.

▶ **Theorem 17.** *For an instance with k non-switchable crossings, we can decide whether there is a configuration without self-intersections in $O(k^{k^{k^{k^{\cdots^{k}}}}} \cdot n^2)$ time.*

Proof. Suppose we have an arrangement $\mathcal{A}$ for which the answer is yes, and let G be its (planar) graph. We augment G by adding a new vertex ω that represents the outer face; we create an edge from all vertices on the frame to ω Now consider a configuration K for $\mathcal{A}$ without self-intersections. All curves in $\mathcal{A}$ are now edge-disjoint cycles in G.

Consider the cycles that pass through at least one non-switchable crossing. Because each non-switchable crossing involves only two cycles, there are at most $2k$ such cycles, and we will assign distinct colors to these cycles.

Replace each non-switchable crossing by four terminals corresponding to the four incident edges. Let G' be the resulting graph. This cuts up the cycles that pass through the non-switchable crossings into exactly $2k$ edge-disjoint colored paths that each start and end at terminals. For each non-switchable crossing, the paths incident to opposing terminals have the same color. On the other hand, because any cycle passes through any non-switchable crossing only once, every non-switchable crossing is incident to two distinct colors, so any two of its terminals that are not opposing have different colors.

We wish to recover a configuration K' without self-intersections for G based on only the following information derived from K:

1. the pairs of terminals that are connected by the $2k$ edge-disjoint paths; and
2. for each such pair, the color of the path that connects them.

For a given graph, one can decide the existence of edge-disjoint paths that connect specified pairs of terminals (and report the paths if they exist) in $O(k^{k^{k^{k^{\cdots^{k}}}}} \cdot n^2)$ time [13]. If we apply this algorithm to G' with the above information derived from K, it will return that there exists a set of edge-disjoint paths.

For any such set of edge-disjoint paths, we can correspondingly color the edges of G, and give all the edges that do not lie on any of the $2k$ paths a new color (e.g. color $2k+1$). This results in an Eulerian partition of G that is straight at each of the non-switchable crossings. Any switch will either be incident to four edges of the same color, or two pairs of two edges with the same color. We now define our configuration K' as follows. If the four edges incident to a switch s have the same color, we configure s so as not to create a crossing. If on the other hand, s is incident to two pairs of edges with different colors, we configure s in the obvious way: connecting both pairs of edges with the same color. Now, each strand of K' has a single color. We show that no strand of K' intersects itself. Indeed, any self-intersection must occur at a vertex incident to four edges of the same color. Because non-switchables are not incident to four edges of the same color, the only vertices at which a strand might self-intersect is at a switch. However, by construction, switches whose four incident edges all have the same color are configured not to cross. Hence, configuration K' has no self-intersections. ◀

Again, the proof is constructive; we now have all the ingredients for our main algorithm:

▶ **Algorithm 18.**

- *Construct G' by connecting all boundary vertices to an additional vertex ω on the outer face, and replacing each non-switchable crossing by four terminals.*
- *Call a coloring of the terminals* valid *if it uses at most $2k$ colors, and for each non-switchable crossing, the opposing terminals have the same color, and non-opposing terminals have distinct colors. There are at most $k^{2k(2k-1)}$ valid colorings.*
- *For each valid coloring of the terminals:*
  - *For a given coloring, call a matching between the terminals valid if it matches only pairs of terminals that have the same color. For a given coloring, there are at most $(2k-1)!!$ valid matchings [2].*
  - *For each valid matching:*
    - *Run [13] (in $O(k^{k^{k^{k^{\cdots^{k}}}}} \cdot n^2)$ time).*
    - *If it returns true, store the resulting collection of paths.*
- *If no paths were stored, return false.*
- *Otherwise, set all switches along the stored paths accordingly so that they become curves.*

- *For the remainder, (greedily) cover G' with edge-disjoint cycles. If any cycle has a self-intersections, simply split it into multiple cycles.*
- *Set all remaining switches so that each cycle corresponds to a curve.*

The above ideas can be partially extended to the case of removing popular faces. However, the general question whether removing popular faces is FPT in the number of non-switchable crossings remains open.

4 Conclusion

We have shown that the problem of minimizing either the number of self-intersections or the number of popular faces in a curve arrangement is NP-hard. However, in our application of puzzle generation, we expect the number of non-switchable crossings to be small, and we show that minimizing the number of self-intersections is FPT in the number of non-switchable crossings. This makes our results potentially suitable for generating advanced, rather than expert, curved nonograms, although one remaining question from a practical point of view is whether the dependency on k can be improved. On the other hand, the problem of minimizing the number of popular faces remains open, which means we cannot yet (efficiently) generate basic curved nonograms using this method.

References

1 Kees Batenburg and Walter Kosters. On the difficulty of nonograms. *ICGA journal*, 35:195–205, December 2012. doi:10.3233/ICG-2012-35402.

2 David Callan. A combinatorial survey of identities for the double factorial. https://arxiv.org/abs/0906.1317, 2009.

3 Yen-Chi Chen and Shun-Shii Lin. A fast nonogram solver that won the TAAI 2017 and ICGA 2018 tournaments. *ICGA Journal*, 41(1):2–14, 2019. doi:10.3233/ICG-190097.

4 Phoebe de Nooijer. Resolving popular faces in curve arrangements. Master's thesis, Utrecht University, 2022. URL: https://studenttheses.uu.nl/handle/20.500.12932/494.

5 Phoebe de Nooijer, Soeren Nickel, Alexandra Weinberger, Zuzana Masárová, Tamara Mchedlidze, Maarten Löffler, and Günter Rote. Removing popular faces in curve arrangements. In *Proc. 31st International Symposium on Graph Drawing*, 2023.

6 Phoebe de Nooijer, Soeren Terziadis, Alexandra Weinberger, Zuzana Masárová, Tamara Mchedlidze, Maarten Löffler, and Günter Rote. Removing popular faces in curve arrangements. *Journal of Graph Algorithms and Applications*, 28(2):47–82, November 2024. doi:10.7155/jgaa.v28i2.2988.

7 Cole A. Giller. A family of links and the Conway calculus. *Transactions of the American Mathematical Society*, 270(1):75–109, 1982.

8 Jim Hoste. The Arf invariant of a totally proper link. *Topology and its Applications*, 18(2–3):163–177, 1984.

9 Vaughan F. R. Jones. On knot invariants related to some statistical mechanical models. *Pacific journal of mathematics*, 137(2):311–334, 1989.

10 Louis H. Kauffman. State models and the Jones polynomial. *Topology*, 26(3):395–407, 1987.

11 Louis H. Kauffman. Gauss codes, quantum groups and ribbon Hopf algebras. *Reviews in Mathematical Physics*, 5(4):735–773, 1993.

12 Maarten Löffler, Günter Rote, Soeren Terziadis, and Alexandra Weinberger. On solving simple curved nonograms. In *Proc. 36th International Workshop on Combinatorial Algorithms (IWOCA 2025)*, 2025.

13 Neil Robertson and Paul D. Seymour. Disjoint paths—a survey. *SIAM Journal on Algebraic Discrete Methods*, 6(2):300–305, 1985. doi:10.1137/0606030.

14 Alexander Schrijver. On the uniqueness of kernels. *Journal of Combinatorial Theory, Series B*, 55(1):146–160, May 1992. doi:10.1016/0095-8956(92)90038-Y.

15 Nobuhisa Ueda and Tadaaki Nagao. NP-completeness results for NONOGRAM via parsimonious reductions. Technical Report TR96-0008, Department of Computer Science, Tokyo Institute of Technology, 1996. CiteSeerX 10.1.1.57.5277, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.57.5277.

16 Mees van de Kerkhof, Tim de Jong, Raphael Parment, Maarten Löffler, Amir Vaxman, and Marc J. van Kreveld. Design and automated generation of Japanese picture puzzles. *Comput. Graph. Forum*, 38(2):343–353, 2019. doi:10.1111/cgf.13642.

17 Jan van Rijn. Playing games: The complexity of Klondike, Mahjong, nonograms and animal chess. Master's thesis, Leiden Institute of Advanced Computer Science, Leiden University, 2012.

Show Me Your Best Side: Characteristics of User-Preferred Perspectives for 3D Graph Drawings

Lucas Joos
University of Konstanz, Germany

Maximilian T. Fischer
University of Konstanz, Germany

Falk Schreiber
University of Konstanz, Germany
Monash University, Melbourne, Australia

Karsten Klein
University of Konstanz, Germany

Gavin J. Mooney
Monash University, Melbourne, Australia

Daniel A. Keim
University of Konstanz, Germany

Helen C. Purchase
Monash University, Melbourne, Australia

Abstract

The visual analysis of graphs in 3D has become increasingly popular, accelerated by the rise of immersive technology, such as augmented and virtual reality. Unlike 2D drawings, 3D graph layouts are highly viewpoint-dependent, making perspective selection critical for revealing structural and relational patterns. Despite its importance, there is limited empirical evidence guiding what constitutes an effective or preferred viewpoint from the user's perspective. In this paper, we present a systematic investigation into user-preferred viewpoints in 3D graph visualisations. We conducted a controlled study with 23 participants in a virtual reality environment, where users selected their most and least preferred viewpoints for 36 different graphs varying in size and layout. From this data, enriched by qualitative feedback, we distil common strategies underlying viewpoint choice. We further analyse the alignment of user preferences with classical 2D aesthetic criteria (e.g., *Crossings*), 3D-specific measures (e.g., *Node-Node Occlusion*), and introduce a novel measure capturing the perceivability of a graph's principal axes (*Isometric Viewpoint Deviation*). Our data-driven analysis indicates that *Stress*, *Crossings*, *Gabriel Ratio*, *Edge-Node Overlap*, and *Isometric Viewpoint Deviation* are key indicators of viewpoint preference. Beyond our findings, we contribute a publicly available dataset consisting of the graphs and computed aesthetic measures, supporting further research and the development of viewpoint evaluation measures for 3D graph drawing.

2012 ACM Subject Classification Human-centered computing → Empirical studies in visualization; Human-centered computing → Graph drawings

Keywords and phrases Graph Aesthetics, Immersive 3D, Node-Link Diagrams, Empirical Evaluation

Digital Object Identifier 10.4230/LIPIcs.GD.2025.37

Related Version *Full Version*: `https://arxiv.org/abs/2506.09212`

Funding The authors gratefully acknowledge financial support by the Federal Ministry for Economic Affairs and Climate Action (BMWK, grant No. 03EI1048D) and the Deutsche Forschungsgemeinschaft (DFG) – Project-ID 251654672 – TRR 161.

1 Introduction

Beyond traditional graph drawing in two-dimensional space, the rise of immersive technology has enabled numerous applications and studies where graphs are arranged and viewed in three-dimensional space using devices that support stereoscopic 3D (S3D) vision [20]. Virtual (VR) and augmented reality (AR) devices are increasingly employed for this purpose, due to their growing availability, quality, and affordability, across fields such as biology [25], neuroscience [34], and the social sciences [11]. Although rare approaches visualise graphs



© Lucas Joos, Gavin J. Mooney, Maximilian T. Fischer, Daniel A. Keim, Falk Schreiber, Helen C. Purchase, and Karsten Klein;
licensed under Creative Commons License CC-BY 4.0



as matrices in 3D space [33], the common method remains node-link diagrams [21] where node positions are typically derived from layout algorithms or reflect semantic 3D positions (such as brain regions [18]). Under certain conditions, S3D environments have been shown to enhance graph-related tasks, such as community detection [15] and path tracing [28].

However, due to the three-dimensional representation and the impact of depth and occlusion, viewpoints might either facilitate or prohibit the identification of important aspects and structures, thereby supporting or hindering the analysis process. Therefore, 3D visualisations are often viewed from several viewpoints for interactive graph analysis, which is mostly done by the analyst in an ad-hoc fashion. Investigating the impact of different viewpoints and their potential relation to layout characteristics and quality measures consequently deserves increased attention. One pathway to derive insight on this impact is to investigate the variability of established 2D quality measures for a sample of viewpoints without considering human perception at all. However, these measures might not be a good fit for the quality of perceived structures in a 3D environment. A complementary approach is to investigate the characteristics of viewpoints chosen by humans.

It is well-known that preference for a visualisation does not necessarily strongly correlate with better task performance or lower cognitive load [16]. Still, preferences that are not solely based on aesthetic appearance, but motivated by a context in graph analysis, can guide the evaluation of quality measures and facilitate the search for new ones. Even if humans would sometimes prefer less advantageous viewpoints, it is an important goal to avoid viewpoints that are reasoned to be subpar, in particular if objective guidance for improvement can be given. In interactive analysis, the choice of what to explore next can be influenced, or even misled, by the previous choice, especially if that earlier choice did not provide enough information to make reasonable decisions going forward. Hence, optimised initial viewpoints can be especially valuable. Further, identifying characteristics that lead to specific preferences allows us to gain further insight into human graph analysis behaviour. In the literature, measures intended to capture the quality of a graph drawing are described as either *aesthetic* or *readability* measures, terms whose meanings partly overlap but do not always coincide. As our study centres on measures grounded in viewers' preferences, we adopt the term *aesthetic* throughout this paper. Our investigation aims to better understand human preference for 3D viewpoint selection, investigating the following research question:

RQ: *What are users' preferences regarding viewpoints in 3D graph drawings, and to what extent can individual aesthetic measures (e.g., edge crossings, stress) or combinations of these characterise preferred or unfavoured perspectives?*

Thereby, we make the following contributions:

- A VR **user experiment** ($n = 23$) collecting a set of user-preferred and -disregarded perspectives for a broad set of 36 3D graph drawings along with qualitative user feedback.
- A detailed analysis of the study results, consisting of a **qualitative and quantitative analysis** evaluating the expressiveness of individual aesthetic measures (21 in total, including one introduced in this work) and **combinations** of these.
- We publish our **study data** along with all results and analyses as **open-access**, providing a valuable dataset for future research on this topic: 10.18419/DARUS-5108 [22].

2 Related Work

There is some evidence that people, when presented with a static graph drawing, tend to prefer those with lower values for quality measures such as edge crossings or stress [4]. However, in interactive settings, they do not necessarily optimise for the lowest values [26].

With the additional degree of freedom in 3D visualisations, this tendency may be even more pronounced due to both the limited applicability of classical quality measures to viewpoint evaluation and the difficulty humans face in judging layout quality for analytical purposes. Over the years, many quality measures have been proposed [3, 8], some of which are better supported by empirical studies than others. A relevant question is to what extent viewers can actually perceive and distinguish different values of such measures, i.e., whether this information can help them identify better viewpoints. Mooney et al. investigated the perception of stress [31], asking participants to determine which of two drawings had higher stress. They found that the task was feasible above a certain threshold difference.

In 2D settings, Huang et al. demonstrated that optimising a combination of criteria can improve task performance [17]. Several methods have been developed to optimise multiple quality measures simultaneously, or to support this through interactive user involvement [39, 37, 6]. Since layouts can either facilitate or hinder task performance, they also influence cognitive load [48, 16, 45], which is another consideration for future work in our context. Closely related to our approach of exploring quality measures through user preferences are those that invert the classical "algorithm first, evaluation second" paradigm, deriving insight from human-generated layouts instead [23, 36].

In 3D graph visualisation, Joos et al. computed aesthetic measures for a large number of sampled viewpoints and visualised these in a VR environment, helping users identify potentially beneficial perspectives, which received positive feedback from domain experts [19]. Similarly, Wageningen et al. investigated how 2D quality measures vary across viewpoints sampled on a sphere surrounding 3D graph layouts, and applied gradient descent methods to find views with the highest aesthetic values [43, 44]. The investigation of the impact of such measures in stereoscopic 3D is more complex than in 2D, as factors such as depth, distance, and perspective might influence the perception significantly, as e.g., for edge crossings [49]. Drogemuller et al. [10] studied edge curvature aesthetics in Lombardi-inspired 3D layouts within stereoscopic 3D, finding that straight edges outperformed curved ones. Bennett et al. and Ware et al. have also examined the perceptual foundations of graph visualisation, including links to Gestalt principles [3, 45].

Although considerable research has addressed drawing principles for 2D graph visualisations, and some initial work has explored advantageous viewpoints in 3D drawings, there remains no empirical evaluation of user-preferred perspectives in 3D and whether these preferences can be characterised by existing aesthetic measures or require new ones.

3 Methodology

To examine how viewers choose preferred viewpoints in 3D graph drawings, and to what extent these can be explained by aesthetic measures (see Section 1), we conducted an empirical user study and analysed the data through a structured pipeline (see Section 4). Participants were asked to rotate a series of 3D graphs, and for each, to identify best and worst viewpoints. In the following, we describe the dataset, the VR prototype, the study procedure, the aesthetic measures, and participant details.

3.1 Stimuli

Drawing on experience from earlier VR studies, we aimed to keep headset time below 45 minutes. With an average of one minute per graph plus setup and calibration, we decided on a sample of 36 graphs. To support general and class-specific conclusions, we included graphs from four size classes (`S-S` (20 nodes), `S-M` (50), `S-L` (100), and `S-XL` (200)) and three layout

types (energy-based (`L-E`), layered (`L-L`), and semantic (`L-S`) types), aligned with those typically used in immersive settings [20]. With three graphs per size-layout combination, this yields 36 graph drawings. Edge density varied between 0.75% and 33.3%, computed as $\frac{2m}{n(n-1)}$ for a graph $G = (V, E)$, $|V| = n$, $|E| = m$. Figure 2 shows a sample of these graphs, while a full documentation can be found in [22] and the full version. Following, we describe each class and how the 3D graph drawings were created.

Energy-Based Layouts (L-E)

Energy-based algorithms are widely used for immersive graph visualisations, particularly when node positions lack semantic meaning, as many drawing algorithms natively support 3D. For this layout type, we selected 12 graphs (three per size) and computed 3D layouts using the *Stress Minimisation* algorithm from OGDF [5]. Graphs were drawn from the *Rome* [9] dataset and *SuiteSparse Matrix Collection* [7], both standard benchmarks. To include sparser, structured samples, we created graphs with the OGDF Random Tree Generator.

Layered Layouts (L-L)

Layered graphs are relevant in cases with hierarchical structures or distinct node types [27]. We followed a method from a recent VR study on layered graphs for generation [14]. We used two layers for `S-S` and `S-M` and three for `S-L` and `S-XL`. Given a target size n, nodes were partitioned into l layers with a maximum deviation of $\pm 10\%$ from $\frac{n}{l}$, ensuring a roughly even split with some perturbation. Within each layer, edges were added using OGDF's *Simple Connected Graph Generator*, followed by the *FMMM* layout algorithm for 2D node placement. Between adjacent layers, 5 to $0.7n$ inter-layer edges were added at random to produce varying densities.

Semantic Layouts (L-S)

A key advantage of 3D visualisations is the ability to place nodes at meaningful spatial, semantic locations. A prominent case is brain activity networks, where nodes represent brain regions (centroids) and edges indicate functional correlations [18, 34]. These networks are undirected, fully connected, and weighted. Applying a threshold removes weaker edges, and discarding the weights yields an unweighted graph that highlights strong correlations. Region granularity is adjustable via brain atlases, offering flexibility in node and edge counts. Given their frequent use in immersive contexts, meaningful spatial structure, and adaptability, we included such networks in the evaluation. For `S-L` and `S-XL`, we used human fMRI data from the Human Connectome Project [41, 42] with the Schaefer (100 regions) and Brainnetome (246 downsampled to 200) atlases [38, 12]. For `S-S` and `S-M`, we used rat fMRI data [1, 2] (53 regions, downsampled to 20 or 50). Based on prior experience, edge thresholds were tuned to yield realistic connectivity for fMRI analysis (see Table 2).

3.2 Application

We developed a VR application for the Apple Vision Pro (2024 release) using Unity. The headset offered high resolution, substantial processing capabilities, and native hand- and gaze-based interaction, making it ideal for this evaluation. The virtual environment consisted of a minimal room designed to provide spatial orientation while minimising distractions. Participants experienced the environment while seated at a fixed position (only head movement was allowed). Graphs with pre-computed 3D layouts were presented in stereoscopic space,

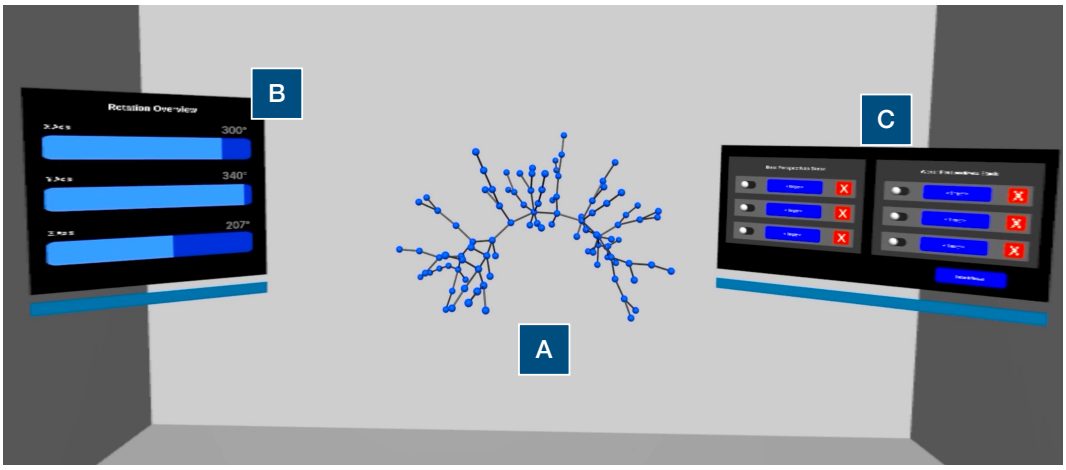


Figure 1 The VR study application, perceived and controlled using the Apple Vision Pro. The 3D graph drawing (A) can be rotated by a gaze and pinch interaction, or sliders (B). Users can store up to three preferred or disregarded perspectives (C).

using blue spheres for nodes and black tubes for edges – a typical representation for graphs in immersive settings [20] (see Figure 1). Users rotated the graph by directing their gaze at it and performing a pinch gesture (thumb and index finger together), applying the hand rotation to the graph. Three sliders provided fine-grained rotation control around the X, Y, and Z axes. An interactive panel allowed participants to store up to three viewpoints per category (best or worst), review them, and select one final view for each category using toggles. Submissions were only enabled after both selections were made. Upon submission, the selected views and associated metadata (e.g., task duration and graph position) were logged in a database. Both user interface panels were movable within the 3D environment to avoid occlusion and support ergonomic interaction. While graphs remained fixed during each task, participants could adjust the default placement of the visualisation between tasks by moving a bounding sphere. Graph size was fixed, and full visibility was ensured at all times.

3.3 Study Procedure

The study took place in a controlled university lab and lasted about one hour. Participants received 15 € compensation. After providing informed consent, participants watched a four-minute video explaining the setup, application, and task. They were instructed to rotate each graph and identify both the *best* and *worst* viewing perspectives from their point of view. Participants were informed that perspectives were not only to be evaluated based on visual appeal, but also on their ability to convey information relevant to task-solving. After addressing any questions, the headset was fitted and calibrated. A screen-mirroring setup allowed the instructor to monitor the session and offer guidance if needed. Participants completed two trial tasks to familiarise themselves with the interface and determine a default graph position. In the main study, they evaluated 36 graphs shown in random order. Submissions were only possible after rotating the graph at least 180° along each axis ensuring a thorough exploration. A soft time limit of one minute per task was set, which was not enforced, but participants were reminded if they exceeded it significantly. Breaks were allowed at any time, and questions were addressed as they arose. Audio was recorded, and participants were encouraged to verbalise their strategies. At the end, participants completed a brief questionnaire on demographics, strategies, and their experience before leaving.

3.4 Aesthetic Measures

Our analysis uses 21 aesthetic measures, mostly established 2D graph drawing metrics grounded in longstanding principles and empirical studies [30, 3, 35]. Moreover, we include 3D-specific overlap measures and a novel metric for structural perceptibility in 3D.

2D Measures

To obtain 2D positions for aesthetic evaluation, we applied perspective projection to the 3D graph from a given viewpoint. We used the Python library `GdMetriX` [32], which implements all relevant 2D measures found in the literature and required for our analysis. The following measures were calculated: *edge crossings* CR, *stress* ST [47] (discrepancy of geometric and topological distances), *crossing angular resolution* CAR (how much deviate angles of crossing edges from an optimum), *bounding box area* AR, *aspect ratio* ASP, *node concentration* CON [40] (estimating how (un)evenly distributed nodes are), *node orthogonality* NO (how "packed" are nodes on the grid), *Gabriel ratio* GR [30] (measuring whether nodes are too close to edges), *angular resolution* ANGR [35] (the angle deviation of incident edges from an optimum), *edge orthogonality* EO [35] (measuring how horizontal/vertical edges are), and *edge length deviation* ELD [30] (the mean deviation of edge lengths, aiming for uniformity). To capture symmetry [29], we included three edge-based measures [24]: *reflective* ESR, *rotational* ESO, and *translational* EST symmetry, measuring whether a symmetry axis exists and quantifying rotational or translational invariance.

3D Overlap

Unlike in 2D, where edge crossings are the main form of occlusion, 3D graph drawings are more susceptible to various overlaps. Prior work highlights their relevance for users [19], yet most studies only consider node-node or edge-node overlaps, without accounting for the order or visual depth. User feedback (see Subsection 4.1) suggests that the direction of occlusion matters. When a node appears in front of an edge, it can falsely imply a connection where none exists, making this type of overlap particularly misleading. Conversely, when an edge passes in front of a node, the node often remains partially visible because edges are typically thinner. However, this can still obscure important parts of the node and lead to misinterpretation, especially in dense regions. Users also noted that the degree of overlap influences perception, not only their pure existence. To reflect this, we include six overlap measures: *node-node* NNO, *edge-node* ENO, and *node-edge* NEO count occurrences, while *node-node overlap area* NNOA, *edge-node overlap area* ENOA, and *node-edge overlap area* NEOA quantify extent. Node-node areas (circle-circle intersections) can be computed in closed form [46]. Overlaps involving rectangles (edges) and circles (nodes) are more complicated and approximated via fine-grained rasterisation.

Isometric Viewpoint Deviation

While previous measures capture various aspects of viewpoint quality, they do not assess whether a graph's global structure is perceivable from a given perspective. This aspect, although difficult to quantify, is crucial for understanding user preferences (see Subsection 4.1). It is not always clear what the overall structure of a graph is, or how to determine whether a viewpoint supports its perception, especially for arbitrary graphs with no prior semantic knowledge. However, user feedback in our study provided valuable insight into how participants perceive structure. Many described preferring a "diagonal view" that offers

a balanced look along the graph's main axes, avoiding alignment between those axes and the view direction. For most graphs, the perceived axes correspond to directions with high node spread, which can be estimated using principal component analysis (PCA). Inspired by Farish's concept of *isometric projections* [13], established in the 19th century, which are views in which the axes appear equally foreshortened (120° angle between each pair), we define a measure to quantify deviation from this ideal, taking into account whether PCA-derived axes meaningfully describe the graph's structure (*anisotropy*). Let the normalised view vector be $\boldsymbol{v} \in \mathbb{R}^3$, and the PCA of the node positions yield orthonormal eigenvectors $\boldsymbol{E} = [\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3] \in \mathbb{R}^{3\times 3}$ and corresponding eigenvalues $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \lambda_3]^\top$. We define the *Isometric Viewpoint Deviation* score as $\mathrm{ISO} = 1 - \alpha \cdot \frac{\sigma_w}{\sigma_{\max}}$ with

- $\boldsymbol{a} = \left|\boldsymbol{E}^\top \boldsymbol{v}\right|$: absolute projections on principal axes,
- $\boldsymbol{a}_{\mathrm{norm}} = \boldsymbol{a} / \sum a_i$: normalised projection weights,
- $\mu = \sum w_i \, a_{\mathrm{norm},i}$: weighted mean,
- $\sigma_w = \sqrt{\sum w_i \, (a_{\mathrm{norm},i} - \mu)^2}$: weighted standard deviation of projections,
- $\alpha = \sigma(\boldsymbol{w})/\sigma_{\max}$: normalised anisotropy factor,
- $\sigma_{\max} = 1/\sqrt{3}$: normalisation constant.

The anisotropy factor α ensures the score only penalises unbalanced projections when the graph has meaningful, deviating main axes (i.e., uneven eigenvalues). It is zero for isotropic graphs and one for maximally anisotropic ones. This is combined with σ_w, which captures how unevenly the principal axes are represented in the current view, normalised by the worst-case imbalance. The resulting score ISO ranges from 0 to 1, where 0 indicates highly imbalanced views of strongly structured graphs, and 1 corresponds to either balanced views or graphs lacking dominant directions. While we acknowledge that there may be other, potentially better or more sophisticated measures to quantify the perception of main axes or to define *structure*, both the tests and user-reported strategies suggest that this approach is worth incorporating into the analysis and can serve as a baseline for future methods.

3.5 Participants

We recruited 23 participants (9 female, 14 male) from among students and associates at our university. The average age of participants was 25.96 years (SD = 3.94), with a median age of 25. All participants reported normal or corrected-to-normal vision through contact lenses, as the VR device can not be used with glasses. Participants rated their experience with graphs on a five-point Likert scale ranging from 1 (No Experience) to 5 (Expert Level Experience), resulting in a mean rating of 2.52. Their experience with immersive technologies (such as augmented or virtual reality) was assessed using the same scale, with a mean rating of 2.27. After completing the study, participants rated the task difficulty on a five-point Likert scale from 1 (Very Easy) to 5 (Very Difficult), yielding a mean difficulty rating of 2.17.

4 Results

In the following, we present the qualitative feedback provided by participants, examine how their reported strategies align with their selected viewpoints, and analyse how these preferences vary. We also evaluate the aesthetic measures applied in the study and assess the extent to which individual or combined measures explain users' choices. Numbers in parentheses indicate how participants mentioned a strategy or observation.

4.1 Qualitative Results

During and after the study, participants described their strategies for identifying the most and least preferred viewpoints. Across all graph types, a central theme was ensuring the overall structure was visible ($n = 17$), even at the cost of some overlap or edge crossings. "Diagonal" views on the graph were considered beneficial for this purpose ($n = 3$). Minimising node and edge overlap was frequently mentioned ($n = 5$), with some participants focusing specifically on node-edge overlap ($n = 5$), and one on edge-node overlap. Others stressed the importance of distinguishing connected from unconnected nodes ($n = 1$), and warned against heavy node overlap ($n = 3$), especially in depth ($n = 1$), where structure became hard to discern otherwise ($n = 2$). An observed strategy involved first finding a view revealing the global structure, then refining the view to reduce overlaps and crossings ($n = 3$). Generally, global understanding was prioritised over resolving all local ambiguities. Participants identified several detrimental factors: edge crossings, compressed visualisation (especially into depth), misleading projections, and excessive flattening into 2D. Wide layouts were preferred over tall ones, as they better matched the VR field of view. Some noted that misleading visual cues, such as nodes occluding edges, suggesting false connections ($n = 4$), were problematic, as were edges overlapping nodes ($n = 2$). However, one participant mentioned that partially occluded long edges were mostly acceptable, as their course could still be inferred (Gestalt principle of continuity). Partial visibility of nodes or edges was sometimes sufficient for comprehension, while a larger projected area appeared to support understanding ($n = 2$).

For layered graphs, a "tilted side view" was frequently communicated as most effective ($n = 9$), while "orthogonal" views often led to overlap and were less helpful ($n = 7$). Similar comments applied to semantically laid-out graphs, especially where parallel edges aligned with the view axis ($n = 3$). Energy-based layouts prompted more diverse strategies. For tree-like structures, views that aligned the main branch with the view axis were discouraged ($n = 2$), with diagonal views again preferred ($n = 1$). In multi-component graphs (e.g., E-9, see in [22]), overlap among components was a concern ($n = 4$), and placing the more complex part in front improved clarity ($n = 3$). The trade-off between revealing structure and reducing overlap was especially apparent for graph E-2. Participants noted that a slightly diagonal, non-planar view allowed the structure to be perceived, even at the cost of increased overlap ($n = 3$). For E-6, the optimal view appeared task-dependent: diagonal views helped expose internal structure ($n = 3$), while frontal or side views reduced overlap but obscured the overall form ($n = 2$). Individual users sought a view that would reveal the full structure in a way that could be reconstructed by others ($n = 1$), attempted to identify principal components ($n = 1$), or aimed to balance visibility with occlusion ($n = 1$). While selecting the best view was considered challenging in some cases, participants found the 3D environment intuitive and helpful for structural insight. Three participants compared the experience to viewing molecular models, and two noted that 3D perspectives revealed structures that would be difficult to interpret in 2D without stereoscopic cues.

4.2 User-Chosen Perspectives

In the following, we discuss representative user-selected perspectives and their distributions for a sample of eight graphs (see Figure 2). All selected perspectives across users and graph visualisations are shown in [22] and the full version.

For `L-S` graphs, users made relatively consistent selections. In smaller, less dense graphs (e.g., `S-2`), the "optimal" perspectives spread out the graph, minimising occlusion. In "worst" views, the graph appears compact with heavy overlap. In more complex graphs (e.g., `S-9`),

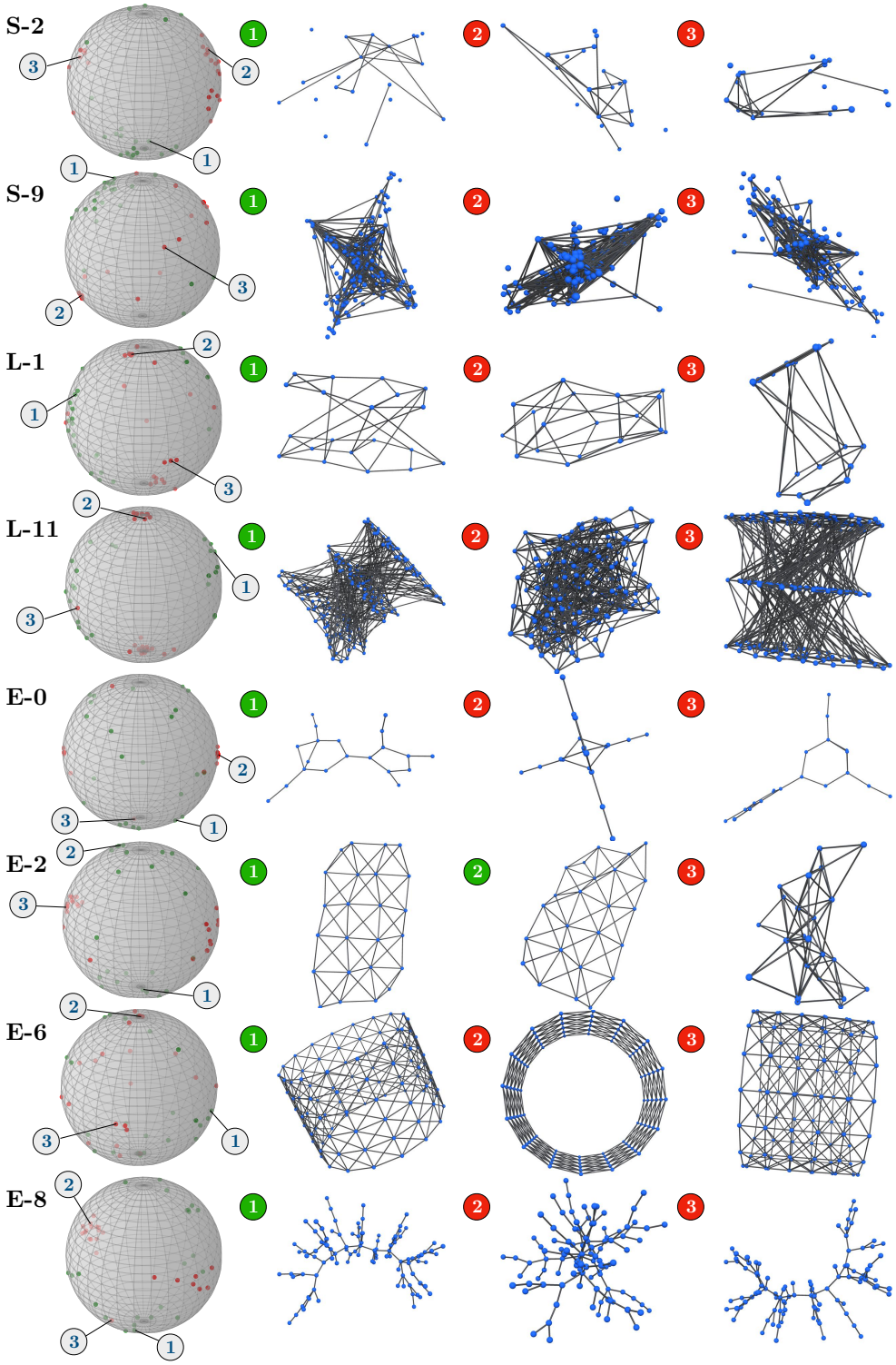


Figure 2 Example of eight graphs (of 36, reflecting the different types) showing the distribution of selected perspectives (sphere surface) along with three perspective projections (either *best* (green) or *worst* (red)), discussed in Subsection 4.2. The complete set of perspectives is shown in [22] and the full version.

"positive" perspectives reveal structure along distinct axes. In contrast, "poor" views align these axes with the viewing direction, causing occlusion, which is consistent with reported strategies. `L-L` graphs (e.g., `L-1`, `L-11`) show similar patterns. "Best" perspectives expose the layered structure, typically via a slightly angled view, matching participant strategies. In "worst" views, a front-on view of one layer hides others, leading to strong overlap. Some users instead viewed layers orthogonally from the side, causing intra-layer node overlap. This was more common in simpler graphs (e.g., `L-1`) and rare for more complex ones (e.g., `L-11`). In `L-E` graphs, perspective selections varied more due to differing topologies despite identical layout algorithms. Structure remained a key factor. In `E-0`, diverse "good" perspectives still conveyed the structure well. For "negative" views, participants often selected misleading angles, e.g., cross-shaped (`2`) or partially obscured (`3`) structures. In `E-2`, a clear 2D projection (`1`) was rarely chosen. Instead, users preferred a less optimal projection (`2`) in terms of crossings but clearer 3D form, as supported by verbal feedback. "Worst" perspectives failed to show the structure and caused major overlap. In `E-6`, resembling a cylinder, users varied more. Many chose a top-down, angled view (`1`) as "optimal". Others preferred (`2`), which had fewer crossings but poorer structural clarity. Those who did not select it as "worst" chose a side-on orthogonal view with severe occlusion. In tree-like graphs (e.g., `E-8`), "good" perspectives revealed the trunk clearly (`1`), while "bad" views rotated it into the image plane (`2`). A few alternatives showed overlapping side branches (`3`).

Overall, the strategies communicated by the participants are clearly reflected in the results. Some graphs prompted highly consistent choices, while others showed broader variation.

4.3 Aesthetic Measures

Before calculating the aesthetic measures, we converted each chosen perspective's graph position and rotation, along with the user position and rotation, into a comparable view vector representation. We then computed 2D graph projections using a perspective projection matching the VR headset's camera parameters. In addition to node and edge positions, the projected dimensions of the representation objects (circles and rectangles) were calculated. To ensure comparability, all measures were adapted such that 0 indicates lowest and 1 highest

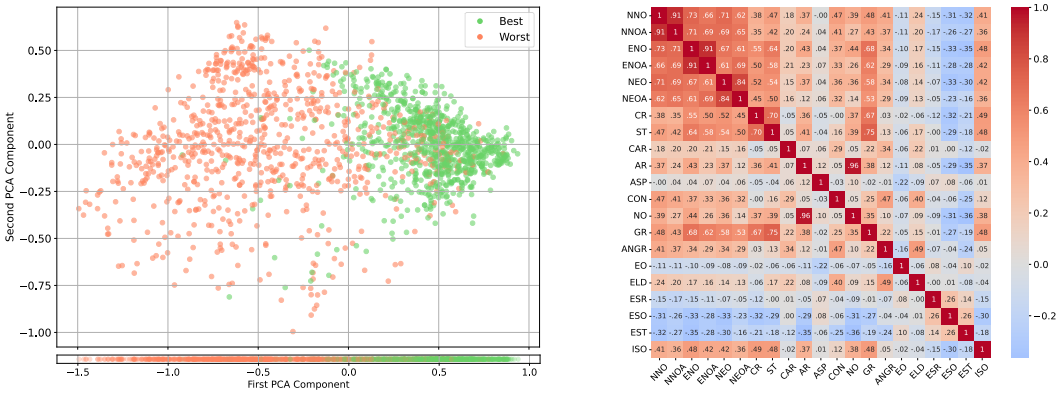


Figure 3 The first two components of a PCA of the *best* (green) and *worst* (red) user-chosen perspectives (left), in total $2 \times 36 \times 23 = 1.656$ samples with 21 measures each. The projection on the first component (left, below) already shows a good separation of the data points. A correlation analysis indicates that some aesthetic measures are highly correlated (especially the overlap measures NN*, EN*, and NE*), while others are not (right).

quality. For each selected perspective (best and worst), we computed the aesthetic measures introduced in Subsection 3.4. Additionally, we uniformly sampled 5000 perspectives per graph using *Fibonacci Lattice* and calculated all aesthetic measures for these as well. This enabled us to determine the full range of achievable values and normalise the selected perspectives accordingly. The resulting scores indicate where each selected perspective falls within this range. Figure 4 presents score distributions across all measures and selected perspectives, and Table 1 summarises mean scores across all data, layout types, and sizes (see Table 2 for graph-level results). The findings suggest that several measures, i.e., CAR, ASP, CON, ANGR, EO, ELD, ESR, ESO, and EST, do not characterise user choices. In contrast, the six overlap measures, along with CR, ST, NO, GR, and ISO, appear more relevant, as they differentiate best and worst perspectives to a certain extent. A user-centric analysis of the aesthetic measure results did not reveal any outliers or substantial user-specific anomalies. A correlation analysis (Figure 3, right) shows that the overlap measures are highly correlated (particularly overlap counts with corresponding areas, which is not surprising). Other notable correlations include those between NO and AR, and among CR, ST, and GR. In Subsection 4.4, we further examine how well individual and combined measures explain user preferences.

4.4 Combinations of Aesthetics

Based on the calculated aesthetic measures, we investigate how well they align with user judgements and whether combined measures can better reflect their importance. A combined measure approximating user choices could be valuable in supporting the automated selection of *optimal* viewpoints. To get an initial impression of the data interdependences, we performed a PCA on the study results (see Figure 3, left). Interestingly, although not designed for this purpose, the first component already provides a good separation between best and worst perspectives. Since each PCA component represents a weighted linear combination of the original features (here, aesthetic measures), we were motivated to explore interpretable, linear combinations of aesthetics that match the collected data. Given these first insights, and considering both the interpretability and the typically lower risk of overfitting associated with linear models – particularly in settings with limited data – we

Table 1 Mean aesthetic values (*best* ↑, *worst* ↓) for all graphs (`All`), layout classes `L-S` (*semantic*), `L-L` (*layered*), and `L-E` (*energy-based*), and size classes `S-S` (20 nodes), `S-M` (50), `S-L` (100), and `S-XL` (200). In addition to 21 aesthetic measures, combined scores (RdYlGr colour scale) using logistic regression (C-LR) and sequential quadratic programming (C-SQP) are shown (5 aesthetics each).

Data		NNO	NNOA	ENO	ENOA	NEO	NEOA	CR	ST	CAR	AR	ASP	CON	NO	GR	ANGR	EO	ELD	ESR	ESO	EST	ISO	C-LR	C-SQP
All	↑	.89	.91	.89	.89	.85	.83	.80	.79	.44	.49	.51	.72	.74	.66	.71	.38	.59	.46	.39	.33	.57	.77	.75
	↓	.63	.69	.55	.62	.59	.62	.45	.39	.41	.28	.49	.64	.53	.28	.64	.41	.61	.49	.53	.46	.30	.47	.39
L-S	↑	.90	.90	.89	.89	.81	.79	.76	.81	.51	.55	.52	.73	.81	.63	.71	.43	.76	.51	.38	.33	.57	.76	.74
	↓	.58	.61	.51	.59	.62	.64	.55	.43	.39	.25	.43	.58	.51	.28	.62	.45	.58	.53	.49	.51	.32	.49	.42
L-L	↑	.91	.94	.90	.90	.88	.87	.77	.70	.46	.45	.60	.72	.70	.60	.77	.36	.35	.43	.39	.28	.55	.77	.71
	↓	.78	.81	.68	.69	.66	.67	.44	.47	.45	.32	.48	.71	.56	.30	.73	.37	.71	.44	.48	.39	.27	.52	.44
L-E	↑	.88	.88	.87	.87	.86	.83	.86	.86	.36	.47	.43	.69	.72	.75	.65	.37	.67	.45	.40	.39	.59	.78	.80
	↓	.54	.64	.47	.58	.49	.54	.36	.28	.39	.29	.56	.62	.52	.27	.56	.39	.55	.52	.64	.47	.30	.41	.33
S-S	↑	.98	.99	.96	.96	.92	.92	.84	.82	.42	.50	.51	.75	.80	.63	.78	.37	.72	.33	.28	.26	.59	.82	.77
	↓	.57	.68	.53	.65	.50	.56	.51	.38	.42	.23	.46	.57	.51	.23	.49	.42	.50	.41	.44	.40	.28	.46	.39
S-M	↑	.91	.93	.89	.90	.86	.84	.78	.71	.56	.47	.55	.75	.74	.65	.76	.39	.64	.53	.41	.30	.55	.77	.72
	↓	.64	.68	.56	.61	.55	.60	.52	.41	.38	.29	.46	.61	.55	.37	.65	.43	.59	.51	.51	.43	.36	.50	.45
S-L	↑	.82	.84	.85	.85	.81	.77	.76	.78	.43	.48	.56	.71	.70	.68	.64	.40	.51	.50	.42	.34	.56	.74	.73
	↓	.67	.71	.58	.62	.70	.68	.41	.34	.36	.33	.56	.69	.53	.28	.70	.39	.71	.52	.55	.41	.29	.49	.38
S-XL	↑	.88	.87	.84	.83	.82	.79	.81	.85	.37	.50	.44	.66	.73	.70	.66	.38	.49	.47	.46	.42	.57	.75	.77
	↓	.65	.68	.55	.61	.59	.64	.35	.43	.49	.29	.47	.67	.52	.24	.71	.39	.65	.54	.64	.58	.25	.44	.37

Table 2 For all 36 graphs, we computed the mean values of 21 aesthetic measures and their linear combinations weighted by logistic regression (C-LR) and sequential quadratic programming (C-SQP). Values are shown for participant-favoured (↑) and -disliked (↓) perspectives. The table also lists the number of nodes $|V|$ and edges $|E|$ per graph and is grouped by graph type: *semantic* (S), *layered* (L), and *energy-based* (E).

Graph	|V|	|E|		NNO	NNOA	ENO	ENOA	NEO	NEOA	CR	ST	CAR	AR	ASP	CON	NO	GR	ANGR	EO	ELD	ESR	ESO	EST	ISO	C-LR	C-SQP
S-0	20	16	↑	.92	.95	.96	.96	.88	.86	.74	.77	.74	.51	.56	.78	.81	.64	.67	.32	.76	.32	.29	.09	.46	.76	.72
			↓	.67	.71	.71	.75	.62	.49	.77	.39	.46	.26	.37	.49	.52	.28	.68	.37	.54	.18	.31	.20	.34	.60	.50
S-1	20	22	↑	.96	.98	.94	.95	.84	.88	.80	.75	.43	.68	.58	.75	.91	.37	.80	.39	.82	.30	.45	.38	.64	.80	.70
			↓	.59	.69	.57	.63	.38	.58	.58	.49	.27	.24	.36	.61	.50	.16	.41	.37	.71	.38	.62	.67	.38	.49	.44
S-2	20	19	↑	.96	.98	.93	.92	.82	.81	.76	.90	.41	.49	.60	.76	.86	.56	.74	.42	.83	.32	.36	.16	.36	.72	.72
			↓	.66	.72	.56	.55	.60	.57	.72	.48	.40	.17	.24	.61	.53	.23	.61	.50	.62	.31	.36	.37	.31	.53	.46
S-3	50	100	↑	.92	.89	.93	.90	.91	.91	.83	.82	.50	.62	.67	.82	.92	.69	.78	.48	.73	.68	.28	.32	.65	.83	.79
			↓	.56	.52	.51	.52	.62	.67	.43	.40	.24	.23	.41	.57	.60	.39	.59	.55	.39	.71	.36	.50	.32	.47	.41
S-4	50	100	↑	.90	.93	.94	.93	.87	.82	.92	.79	.73	.59	.56	.64	.87	.77	.81	.46	.85	.47	.26	.18	.69	.85	.83
			↓	.43	.33	.42	.50	.40	.44	.34	.29	.36	.14	.30	.41	.43	.19	.67	.53	.57	.49	.32	.38	.33	.38	.31
S-5	50	100	↑	.86	.88	.87	.89	.80	.86	.61	.80	.68	.59	.64	.80	.89	.63	.85	.41	.92	.67	.55	.28	.70	.75	.72
			↓	.66	.68	.35	.49	.52	.67	.67	.34	.29	.25	.38	.63	.62	.28	.59	.47	.46	.34	.67	.42	.40	.46	.41
S-6	100	144	↑	.77	.81	.83	.84	.84	.72	.79	.76	.57	.45	.55	.61	.64	.53	.76	.42	.63	.62	.30	.45	.56	.75	.70
			↓	.70	.73	.57	.64	.82	.69	.56	.52	.31	.32	.67	.47	.50	.34	.65	.44	.63	.69	.41	.64	.39	.57	.48
S-7	100	137	↑	.77	.84	.87	.89	.83	.75	.77	.79	.45	.47	.47	.77	.70	.72	.48	.60	.80	.26	.47	.45	.63	.76	.76
			↓	.60	.73	.47	.60	.81	.71	.45	.31	.33	.27	.63	.65	.49	.31	.67	.53	.80	.44	.55	.51	.26	.48	.36
S-8	100	131	↑	.83	.75	.89	.89	.63	.59	.82	.75	.51	.58	.38	.75	.73	.75	.68	.38	.71	.68	.40	.30	.58	.74	.76
			↓	.69	.56	.41	.50	.89	.76	.74	.35	.31	.32	.60	.59	.49	.19	.68	.35	.65	.67	.46	.28	.31	.56	.40
S-9	200	194	↑	.95	.92	.91	.87	.73	.77	.71	.95	.52	.56	.36	.60	.79	.74	.68	.38	.76	.82	.78	.38	.51	.73	.78
			↓	.57	.58	.48	.56	.58	.70	.49	.47	.33	.23	.35	.63	.45	.31	.56	.43	.56	.87	.87	.66	.28	.47	.41
S-10	200	202	↑	.96	.93	.82	.85	.81	.75	.53	.77	.22	.53	.50	.80	.78	.42	.56	.51	.40	.76	.19	.48	.48	.67	.61
			↓	.51	.58	.56	.66	.69	.71	.56	.81	.62	.31	.41	.68	.52	.43	.73	.51	.70	.91	.35	.64	.27	.53	.55
S-11	200	150	↑	.94	.93	.82	.83	.81	.81	.89	.91	.35	.52	.34	.73	.79	.77	.73	.37	.91	.18	.27	.47	.58	.75	.81
			↓	.36	.53	.50	.63	.52	.71	.24	.25	.71	.19	.41	.60	.43	.20	.66	.38	.35	.34	.58	.87	.22	.37	.28
L-0	20	36	↑	1.00	1.00	.97	.97	.96	.96	.85	.78	.35	.42	.58	.76	.75	.54	.77	.47	.60	.32	.13	.21	.61	.83	.75
			↓	.73	.78	.65	.74	.65	.72	.56	.38	.32	.20	.61	.51	.48	.19	.56	.49	.48	.38	.27	.32	.21	.51	.40
L-1	20	38	↑	.98	.99	.94	.97	.92	.92	.85	.82	.39	.52	.57	.76	.81	.49	.75	.44	.52	.37	.26	.29	.66	.83	.75
			↓	.71	.75	.64	.73	.62	.63	.62	.61	.35	.28	.45	.58	.53	.31	.51	.44	.63	.42	.33	.48	.34	.55	.51
L-2	20	40	↑	.99	1.00	.93	.95	.90	.93	.65	.63	.49	.42	.65	.75	.68	.53	.78	.35	.43	.39	.26	.21	.49	.74	.65
			↓	.79	.84	.65	.64	.56	.62	.58	.62	.48	.36	.57	.66	.62	.43	.54	.39	.52	.44	.27	.37	.44	.56	.55
L-3	50	101	↑	.94	.97	.90	.92	.90	.89	.72	.53	.59	.43	.52	.72	.67	.56	.82	.38	.36	.43	.39	.32	.54	.76	.65
			↓	.79	.83	.78	.79	.71	.73	.71	.71	.44	.34	.39	.68	.58	.68	.74	.46	.72	.33	.38	.38	.23	.60	.64
L-4	50	102	↑	.96	.98	.90	.94	.91	.91	.69	.44	.51	.40	.67	.80	.72	.59	.74	.20	.39	.48	.51	.32	.50	.74	.62
			↓	.74	.76	.66	.71	.60	.63	.67	.54	.44	.36	.66	.71	.67	.53	.58	.24	.55	.46	.52	.46	.47	.60	.58
L-5	50	93	↑	.98	.99	.93	.93	.94	.92	.65	.54	.70	.38	.54	.82	.62	.60	.81	.48	.38	.58	.30	.24	.28	.70	.61
			↓	.71	.75	.69	.71	.70	.71	.56	.62	.47	.30	.34	.67	.56	.54	.68	.48	.64	.64	.31	.38	.35	.58	.56
L-6	100	207	↑	.85	.89	.86	.85	.85	.77	.76	.77	.11	.40	.55	.72	.60	.51	.78	.27	.26	.66	.61	.19	.49	.74	.68
			↓	.77	.79	.69	.68	.69	.63	.20	.47	.08	.25	.47	.83	.42	.19	.89	.22	.85	.64	.71	.26	.18	.46	.35
L-7	100	253	↑	.86	.92	.88	.88	.86	.86	.63	.66	.70	.44	.73	.75	.67	.65	.80	.17	.31	.14	.38	.15	.56	.72	.68
			↓	.80	.87	.68	.73	.66	.70	.28	.22	.58	.32	.48	.84	.54	.17	.89	.17	.79	.16	.49	.24	.23	.46	.31
L-8	100	284	↑	.86	.92	.93	.86	.86	.84	.86	.90	.32	.60	.67	.66	.79	.75	.68	.49	.18	.73	.39	.38	.58	.81	.81
			↓	.74	.80	.69	.66	.66	.71	.29	.33	.44	.48	.48	.70	.69	.15	.68	.48	.76	.74	.54	.54	.20	.48	.33
L-9	200	431	↑	.90	.92	.87	.84	.89	.82	.81	.78	.33	.48	.54	.74	.68	.70	.86	.27	.30	.07	.23	.28	.68	.79	.77
			↓	.88	.88	.68	.61	.79	.72	.18	.35	.50	.36	.37	.82	.56	.15	.89	.31	.87	.07	.34	.40	.13	.44	.30
L-10	200	487	↑	.81	.86	.82	.84	.81	.80	.88	.73	.55	.49	.64	.59	.73	.74	.72	.37	.21	.82	.62	.24	.55	.76	.75
			↓	.84	.85	.59	.64	.63	.62	.32	.43	.78	.27	.44	.76	.53	.24	.85	.37	.80	.89	.82	.36	.23	.46	.37
L-11	200	508	↑	.77	.83	.87	.82	.81	.82	.86	.84	.48	.42	.52	.60	.69	.57	.76	.42	.19	.10	.60	.49	.61	.77	.75
			↓	.85	.86	.75	.70	.60	.64	.31	.34	.58	.30	.55	.79	.58	.07	.90	.43	.89	.10	.72	.51	.24	.48	.34
E-0	20	22	↑	1.00	1.00	1.00	1.00	.99	.99	.95	.89	.32	.50	.38	.75	.76	.93	.87	.32	.87	.33	.15	.26	.77	.91	.92
			↓	.31	.55	.19	.50	.37	.40	.25	.14	.75	.28	.46	.43	.54	.18	.34	.42	.18	.77	.69	.49	.14	.24	.18
E-1	20	19	↑	.99	1.00	1.00	1.00	1.00	1.00	1.00	.92	.00	.42	.41	.63	.82	.65	.81	.33	.83	.24	.26	.35	.70	.90	.86
			↓	.16	.41	.39	.73	.33	.50	.40	.26	.57	.14	.63	.73	.46	.16	.47	.35	.34	.41	.65	.24	.13	.32	.27
E-2	24	68	↑	.98	1.00	.95	.96	.97	.95	.97	.90	.62	.50	.28	.80	.84	.92	.80	.33	.81	.35	.31	.40	.64	.86	.89
			↓	.51	.68	.39	.57	.42	.52	.14	.10	.16	.17	.47	.50	.44	.14	.26	.42	.43	.37	.49	.48	.25	.31	.20
E-3	48	129	↑	.90	.93	.89	.87	.88	.87	.82	.73	.64	.44	.52	.81	.61	.77	.70	.37	.75	.69	.56	.42	.32	.73	.72
			↓	.70	.77	.66	.64	.57	.64	.55	.44	.37	.28	.60	.59	.44	.41	.43	.37	.41	.71	.62	.45	.49	.58	.51
E-4	50	58	↑	.83	.87	.84	.83	.70	.63	.85	.84	.24	.39	.40	.73	.65	.57	.65	.37	.66	.32	.41	.34	.66	.75	.76
			↓	.70	.77	.56	.58	.53	.53	.44	.29	.25	.37	.48	.70	.61	.23	.83	.39	.70	.43	.56	.47	.47	.50	.39
E-5	50	49	↑	.88	.95	.86	.88	.82	.78	.93	.90	.49	.44	.41	.63	.67	.62	.72	.34	.74	.48	.45	.28	.60	.79	.79
			↓	.51	.73	.39	.55	.34	.39	.30	.09	.57	.30	.59	.56	.48	.11	.77	.33	.88	.48	.82	.46	.18	.31	.21
E-6	96	336	↑	.88	.95	.82	.82	.79	.78	.51	.68	.59	.36	.75	.67	.58	.78	.44	.44	.51	.34	.38	.50	.59	.68	.68
			↓	.66	.80	.72	.72	.69	.71	.35	.42	.66	.28	.58	.64	.46	.66	.60	.45	.66	.35	.48	.47	.44	.55	.52
E-7	100	132	↑	.83	.81	.87	.81	.84	.81	.81	.78	.38	.52	.49	.72	.76	.64	.76	.36	.75	.32	.42	.35	.48	.74	.73
			↓	.59	.62	.64	.57	.66	.68	.44	.24	.28	.40	.53	.70	.65	.23	.41	.39	.35	.26	.59	.32	.36	.52	.37
E-8	100	99	↑	.72	.70	.75	.78	.79	.77	.93	.92	.19	.55	.43	.71	.79	.77	.42	.46	.44	.79	.39	.33	.54	.75	.80
			↓	.47	.47	.35	.48	.44	.48	.35	.17	.23	.32	.62	.80	.56	.28	.83	.47	.91	.78	.67	.44	.27	.36	.28
E-9	198	597	↑	.83	.76	.80	.77	.81	.79	.89	.89	.25	.34	.31	.57	.50	.80	.55	.26	.62	.56	.39	.65	.49	.74	.79
			↓	.69	.66	.52	.53	.52	.56	.47	.78	.38	.32	.34	.49	.48	.38	.76	.28	.52	.55	.58	.65	.36	.47	.51
E-10	200	199	↑	.83	.76	.84	.80	.81	.70	.88	.90	.26	.53	.48	.63	.74	.72	.49	.32	.40	.33	.49	.23	.72	.80	.82
			↓	.67	.64	.48	.53	.56	.53	.40	.26	.27	.41	.61	.62	.63	.22	.71	.32	.73	.35	.65	.26	.33	.44	.33
E-11	209	767	↑	.91	.89	.86	.89	.90	.86	.84	.92	.37	.63	.25	.65	.88	.85	.62	.51	.61	.61	.59	.58	.54	.78	.82
			↓	.46	.55	.35	.58	.41	.56	.18	.16	.23	.21	.76	.66	.49	.19	.34	.53	.47	.74	.85	.87	.19	.30	.21

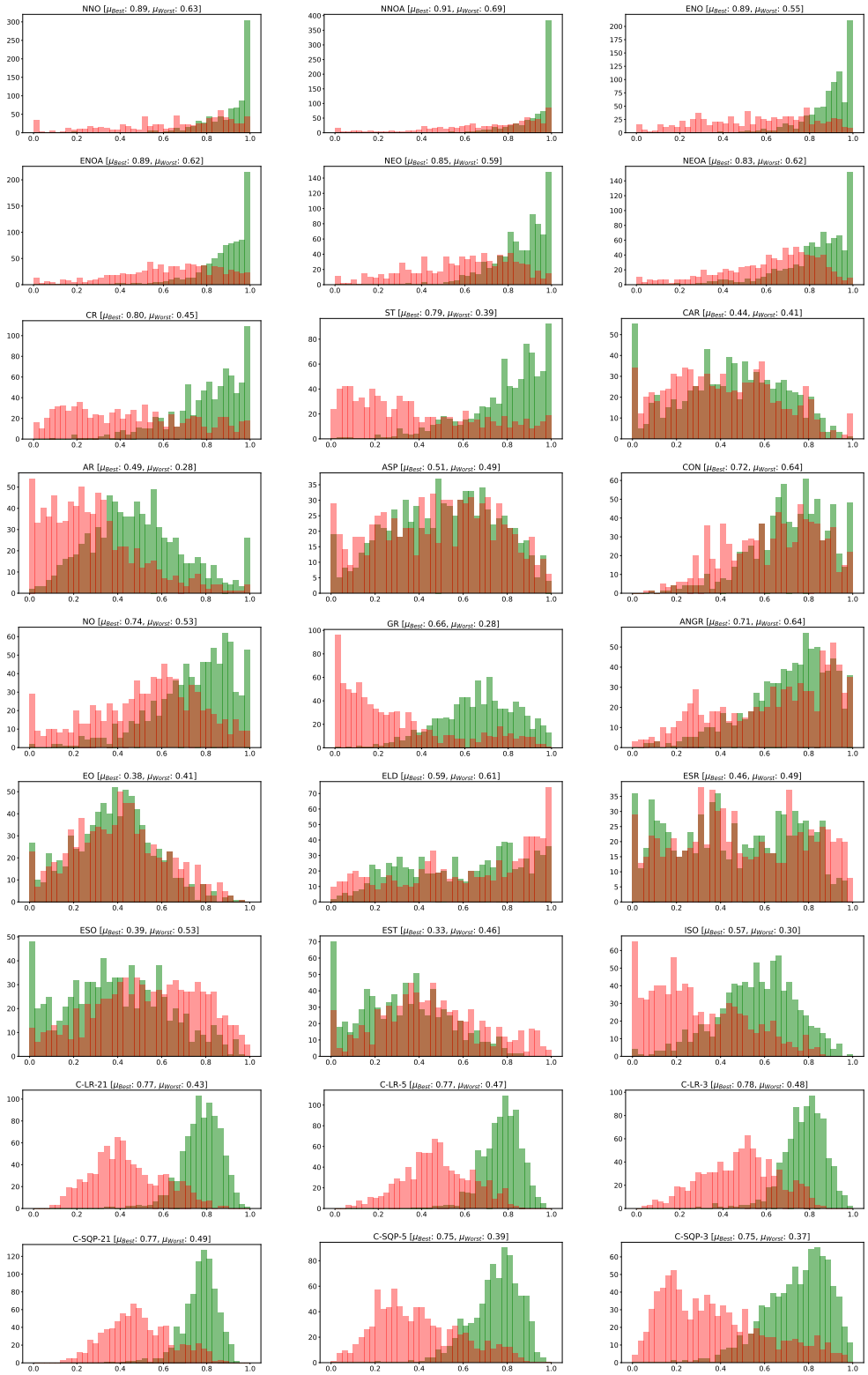


Figure 4 The distributions of values (*best*: green, *worst*: red) for 21 aesthetic measures (top seven rows) and combinations of these (`LR` and `SQP`) with 21, 5, or 3 measures each (last two rows).

aimed to find "optimal" linear aesthetic combinations corresponding to the collected data. We pursued two different approaches for finding such combinations. The first treats the task as a binary classification problem, using *logistic regression* (LR) to distinguish best from worst views. The resulting coefficients can be interpreted as importance weights. However, this approach aims only to classify scores as above or below 0.5, not necessarily to maximise separation. For clearer interpretability, assigning best views scores close to 1 and worst close to 0, we pursued an approach based on *sequential quadratic programming* (SQP): Given aesthetic measures $\{A_1, ..., A_m\}$ with $A_i : p \to v$, where $p \in P$ is a perspective and $0 \leq v \leq 1$, and equally sized classes P_{best} and P_{worst}, we optimise: $\left(\sum_p^n \sum_a^m w_a * A_a(P_{best_p})\right) - \left(\sum_p^n \sum_a^m w_a * A_a(P_{worst_p})\right)$ This could be posed as a linear programme with a convex constraint ($w_a \geq 0$, $\sum w_a = 1$), but such formulations tend to create degenerate solutions with all weights assigned to a single measure. Hence, we instead constrain the weight vector to satisfy $\|\vec{w}\|_2 = 1$ and $w_a \geq 0$, allowing more balanced distributions. This constraint requires a non-linear solving approach, such as SQP, which handles the constraints efficiently and yields optimal weights maximising the separation between best and worst perspectives.

The resulting importance coefficients are shown in Table 3. In addition to computing importances across the entire dataset, we also report results per layout and graph size, and under constraints limiting combinations to five or three measures, to identify the most influential ones. Across conditions, the most relevant measures include the overlap measures, CR, ST, GR, and ISO. Although we expected overlap area to be more important than count, this is not reflected in the results. Interestingly, ENO is typically more relevant than NEO, while NNO appears less expressive. For L-S graphs, overlaps, particularly NNO, along with CR, ST, GR, and AR are most relevant, while ISO is less. In L-L graphs, NEO dominates among overlap measures, with the usual others and especially ISO being important. L-E graphs show similar trends, though ENO is more prominent, while ISO is often excluded. Regarding graph size, smaller graphs appear more affected by overlaps (including NNO), whereas larger ones place more importance on ST and GR. Notably, the SQP method tends to favour measures like ST and GR, where best and worst scores are spread out more, even if some worst views receive high scores. In contrast, LR prefers measures that more clearly separate distributions, such as ISO and the overlap measures, resulting in better classifications (see Figure 4). For comparison with individual aesthetics, we computed linear combinations using weights from both C-LR (logistic regression) and C-SQP (sequential quadratic programming). The results appear to the right in all tables and in Figure 4.

5 Discussion

Our analysis of user-selected viewpoints revealed several user strategies and demonstrated how their choices interrelate with aesthetic measures. Notably, preferences varied across graph types and sizes, often reflecting trade-offs between structural clarity and local features. This supports the view that users do not optimise for a single aesthetic, but rather negotiate among multiple perceptual cues, many of which are not captured by traditional 2D aesthetics.

The strongest 2D aesthetic predictors of viewpoint preference were CR, ST, and GR, aligning with prior findings in 2D [4, 31], and extending their validity into S3D. However, 3D-specific measures, such as ISO and the overlap aesthetics, proved highly relevant. The prominence of ENO over NNO is particularly noteworthy. While one might expect node-node occlusions to be more perceptually disruptive, the results indicate that edge-node intersections may be more misleading. Node-edge occlusions appeared less impactful, possibly due to the Gestalt principle of continuity. Although overlap area was computed as a potentially more accurate representation, it showed no advantage over overlap counts. This may reflect that

Table 3 The relative importance of the aesthetic measures when combined to express the perspective choices of users. We applied logistic regression (`LR`) and sequential quadratic programming (`SQP`) to find the most relevant aesthetic measures when using 21, 5, and 3 measures ($|A|$, i.e. shown as indicated measure cells per each row). We investigated the importance for all data at once (`All`), the three layout classes (`L-S`, `L-L`, `L-E`), and the four graph sizes (`S-S`, `S-M`, `S-L`, `S-XL`).

$\|A\|$	Data	Method	NNO	NNOA	ENO	ENOA	NEO	NEOA	CR	ST	CAR	AR	ASP	CON	NO	GR	ANGR	EO	ELD	ESR	ESO	EST	ISO
21	All	LR	.09	.03	.14	.11	.08	.04	.12	.11	-.02	.11	-.00	.01	.04	.08	.02	.04	-.17	.01	.00	-.02	.19
		SQP	.07	.06	.09	.07	.07	.06	.10	.11	.01	.06	.01	.02	.06	.11	.02						.08
	L-S	LR	.08	.08	.10	.10	.03	.03	.05	.08	.00	.07	.03	.04	.04	.09	.06	.01	.05	.04	-.01	-.05	.08
		SQP	.08	.07	.09	.08	.05	.04	.05	.09	.03	.07	.02	.04	.07	.09	.02		.04				.06
	L-L	LR	-.01	.02	.25	.21	.37	.30	.17	.13	-.05	.12	-.00	-.01	.07	.18	.09	-.04	-.82	-.04	-.01	-.18	.26
		SQP	.05	.05	.08	.08	.09	.07	.12	.09	.00	.05	.04	.00	.05	.11	.02						.10
	L-E	LR	.13	.02	.13	.11	.03	-.02	.10	.19	-.03	.01	-.05	.08	.06	.14	-.04	.06	.02	.02	-.07	.00	.10
		SQP	.08	.06	.09	.06	.08	.06	.11	.13		.04		.02	.04	.11	.02		.03				.06
	S-S	LR	.08	.07	.11	.08	.09	.07	.13	.07	.00	.06	.01	.01	.08	.08	.08	.01	-.06	-.02	.01	-.03	.06
		SQP	.08	.06	.09	.06	.08	.07	.07	.09		.05	.01	.04	.06	.08	.06		.04				.06
	S-M	LR	.06	.09	.17	.16	.14	.04	.13	.06	.08	.11	-.01	-.01	.03	-.06	-.02	.05	-.19	.02	-.02	-.01	.18
		SQP	.07	.07	.09	.08	.08	.07	.07	.08	.05	.05	.02	.04	.05	.07	.03		.01	.01			.05
	S-L	LR	.07	.05	.06	.09	-.03	.01	.11	.20	.05	.04	.01	.05	.05	.11	.03	.04	-.06	.05	-.01	-.09	.17
		SQP	.05	.05	.10	.08	.04	.03	.13	.15	.02	.05		.01	.06	.14		.00					.09
	S-XL	LR	.08	.07	.19	.16	.07	.09	.23	.20	-.13	.07	.04	-.06	.11	.18	-.10	-.02	-.17	-.02	-.08	-.04	.12
		SQP	.07	.05	.09	.07	.07	.04	.13	.12		.06			.06	.13							.09
5	All	LR			.32		.19		.22											.03			.24
		SQP			.19				.20	.23						.22							.16
	L-S	LR		.21		.29			.11			.24											.15
		SQP	.18		.22	.18				.22						.20							
	L-L	LR				.13	.36						.10					.05					.35
		SQP					.17		.24	.17						.22							.20
	L-E	LR			.23	.13				.32						.24		.07					
		SQP			.17		.16		.22	.25						.21							
	S-S	LR			.32		.29		.22			.16							.01				
		SQP	.20		.21		.20			.21						.19							
	S-M	LR	.21		.37						.13					.02							.28
		SQP			.23	.19	.20			.20						.18							
	S-L	LR				.26			.34		.06		.03			.31							
		SQP			.16				.21	.25						.23							.15
	S-XL	LR		.12		.20			.29	.19					.20								
		SQP			.15				.23	.22						.23							.16
3	All	LR			.41		.27																.32
		SQP							.31	.35						.34							
	L-S	LR		.37								.31				.32							
		SQP			.34					.34						.32							
	L-L	LR					.42		.31														.27
		SQP							.36							.33							.31
	L-E	LR			.29				.36							.35							
		SQP							.32	.37						.31							
	S-S	LR	.30		.42				.28														
		SQP			.34		.33			.34													
	S-M	LR			.52						.17												.31
		SQP			.36		.32			.32													
	S-L	LR				.30			.35							.35							
		SQP							.30	.37						.33							
	S-XL	LR			.38				.42						.20								
		SQP							.34	.32						.34							

users prefer views in which objects are clearly distinguishable, as indicated by the relevance of GR penalising nodes and edges in close proximity. Another key insight lies in the differing impact of layout type. Layered and semantic layouts led to more consistent viewpoint choices than energy-based ones. In semantic graphs, where node positions carry intrinsic meaning, participants prioritised preserving perceived structure. This was reflected in the strong relevance of ISO for semantic and layered layouts, where users appeared to prefer perspectives maintaining orthogonal balance or offering diagonal overviews. In contrast, energy-based graphs, which showed greater topological variation in our dataset, elicited more diverse and individual strategies, with user preferences often diverging even within the same graph. These cases suggest that, in the absence of salient or meaningful structure, users default to evaluating local features such as component separation, edge crossings, or occlusions.

One somewhat surprising result was the limited explanatory power of several aesthetics such as ANGR, ASP, and symmetry measures. Although considered beneficial in 2D, these did not significantly distinguish best and worst views in our study. One explanation may be that such aesthetics target fine-grained aspects of local visual quality, while participants were more influenced by coarse, global cues, such as whether the overall structure "popped out" in space, or whether the orientation supported mental model construction. Another possibility is that these aesthetics are harder to judge consistently in stereoscopic views, where depth cues from parallax and motion influence perception in ways not captured by 2D projections.

PCA and correlation analysis revealed that many aesthetic measures are indeed redundant, especially among the overlap measures. Thus, a small number of well-chosen features can explain much of the variance in user preferences. Our comparison of linear combination methods (logistic regression and sequential quadratic programming) confirms that different optimisation objectives yield subtly different rankings, though both approaches consistently identify the most predictive measures. Combinations incorporating ST, GR, CR, ENO, and ISO consistently achieved the best separation between preferred and unfavoured viewpoints.

Beyond the numerical results, qualitative feedback reinforced the value of 3D interaction and stereoscopic cues in disambiguating structure, particularly in dense or ambiguous graphs. Interestingly, user choices were largely consistent, with no substantial differences between novices and more experienced users. This reinforces the value of identifying general principles for viewpoint preference, as we aimed to do in this work.

Despite careful design, this study has limitations. Although we included a broad sample of graphs and layouts, results may differ for other or more specific datasets. Moreover, our participants were primarily novices and generalisation to expert users or larger populations may require further study. While we focused on user *preference*, the relationship to task-solving performance remains unclear. Thus, future work could build on previously introduced methods to identify aesthetic-optimal viewpoints, incorporating our findings on preference-driven combinations to investigate their effects on task performance. Moreover, our analysis has so far treated each viewpoint as static and independent, whereas in real-world use, users navigate through sequences of views. Studying viewpoint transitions, stability, and the temporal dynamics of understanding could offer insights into how users mentally assemble spatial structures over time. Furthermore, based on our tests with the ISO measure, we see potential for developing new, 3D-specific aesthetic measures that reflect how users perceive graphs in a S3D setup. Using our openly accessible dataset, such measures can be evaluated against the user preferences collected in this study.

6 Conclusion

In this work, we conducted a comprehensive study with 23 participants on viewpoint preference in 3D graph visualisation. Participants selected their most and least preferred viewpoints across 36 graphs varying in layout, size, and topology, and shared their selection strategies. Our findings show that, in addition to classical 2D aesthetic measures such as *Stress*, *Crossings*, and the *Gabriel Ratio*, 3D-specific measures, especially *Edge-Node Overlap* and overall structure-capturing methods like our proposed *Isometric Viewpoint Deviation*, are highly reflecting user preference. Participants were often willing to tolerate more overlap and crossings when a viewpoint conveyed the global structure more clearly. Our analysis of combined aesthetics using logistic regression and sequential quadratic programming underlines these results, showing that a small set of measures characterises user choices to a high degree. However, the results indicate the need to adapt and extend traditional quality measures for the 3D context. Beyond our empirical findings, the open-access dataset provides a foundation for future research on viewpoint evaluation and optimisation in immersive graph analysis.

References

1 Sophie Achard, Jean-François Coeurjolly, Pierre Lafaye de Micheaux, Hanâ Lbath, and Jonas Richiardi. Inter-regional correlation estimators for functional magnetic resonance imaging. *NeuroImage*, 282:120388, 2023. doi:10.1016/j.neuroimage.2023.120388.

2 Guillaume J.-P. C. Becq, Emmanuel L. Barbier, Sophie Achard, and Hanâ Lbath. Preprocessed rat brain voxel time series, November 2022. Zenodo. doi:10.5281/zenodo.7254133.

3 Chris Bennett, Jody Ryall, Leo Spalteholz, and Amy Gooch. The Aesthetics of Graph Visualization. In Douglas W. Cunningham, Gary Meyer, and Laszlo Neumann, editors, *Computational Aesthetics in Graphics, Visualization, and Imaging*. The Eurographics Association, 2007. doi:10.2312/COMPAESTH/COMPAESTH07/057-064.

4 Markus Chimani, Peter Eades, Seok-Hee Hong, W. Huang, Karsten Klein, Michael Marner, Ross Smith, and Bruce Thomas. People prefer less stress and fewer crossings. In *Graph Drawing: 22nd International Symposium*, volume 8871 of *LNCS*, pages 523–524, 2014.

5 Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W. Klau, Karsten Klein, and Petra Mutzel. The open graph drawing framework (ogdf). *Handbook of graph drawing and visualization*, 2011:543–569, 2013.

6 Michael K. Coleman and D. Stott Parker. Aesthetics-based graph layout for human consumption. *Software: Practice and Experience*, 26(12):1415–1438, 1996. doi:10.1002/(SICI)1097-024X(199612)26:12<1415::AID-SPE69>3.0.CO;2-P.

7 Timothy A. Davis and Yifan Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software*, 38(1), 2011. doi:10.1145/2049662.2049663.

8 Sara Di Bartolomeo, Tarik Crnovrsanin, David Saffo, Eduardo Puerta, Connor Wilson, and Cody Dunne. Evaluating graph layout algorithms: A systematic review of methods and best practices. *Computer Graphics Forum*, 43(6):e15073, 2024. doi:10.1111/cgf.15073.

9 Giuseppe Di Battista, Ashim Garg, Giuseppe Liotta, Roberto Tamassia, Emanuele Tassinari, and Francesco Vargiu. An experimental comparison of four graph drawing algorithms. *Computational Geometry*, 7(5):303–325, 1997. 11th ACM Symposium on Computational Geometry. doi:10.1016/S0925-7721(96)00005-3.

10 Adam Drogemuller, Andrew Cunningham, James Walsh, James Baumeister, Ross T. Smith, and Bruce H. Thomas. Aesthetics and comprehension of curved 3D graphs in virtual reality. *Journal of Computer Languages*, 75:101198, 2023. doi:10.1016/j.cola.2023.101198.

11 Adam Drogemuller, Andrew Cunningham, James Walsh, William Ross, and Bruce H. Thomas. Vrige: exploring social network interactions in immersive virtual environments. In *Proceedings of the international symposium on big data visual analytics (BDVA)*, volume 5, 2017. doi:10.13140/RG.2.2.34201.67689.

12 Lingzhong Fan, Hai Li, Junjie Zhuo, Yu Zhang, Jiaojian Wang, Liangfu Chen, Zhengyi Yang, Congying Chu, Sangma Xie, Angela R. Laird, Peter T. Fox, Simon B. Eickhoff, Chunshui Yu, and Tianzi Jiang. The human brainnetome atlas: A new brain atlas based on connectional architecture. *Cerebral Cortex*, 26(8):3508–3526, 2016. doi:10.1093/cercor/bhw157.

13 William Farish. On isometrical perspective. *Cambridge Philosophical Transactions*, 1822.

14 Stefan P. Feyer, Bruno Pinaud, Stephen Kobourov, Nicolas Brich, Michael Krone, Andreas Kerren, Michael Behrisch, Falk Schreiber, and Karsten Klein. 2D, 2.5D, or 3D? An exploratory study on multilayer network visualisations in virtual reality. *IEEE Transactions on Visualization & Computer Graphics*, 30(1):469–479, 2024. doi:10.1109/TVCG.2023.3327402.

15 Nicolas Greffard, Fabien Picarougne, and Pascale Kuntz. Beyond the classical monoscopic 3d in graph analytics: An experimental study of the impact of stereoscopy. In *2014 IEEE VIS International Workshop on 3DVis*, pages 19–24, 2014. doi:10.1109/3DVis.2014.7160095.

16 Weidong Huang, Peter Eades, and Seok-Hee Hong. Measuring effectiveness of graph visualizations: A cognitive load perspective. *Information Visualization*, 8(3):139–152, 2009. doi:10.1057/ivs.2009.10.

17 Weidong Huang, Peter Eades, Seok-Hee Hong, and Chun-Cheng Lin. Improving multiple aesthetics produces better graph drawings. *Journal of Visual Languages & Computing*, 24(4):262–272, 2013. doi:10.1016/j.jvlc.2011.12.002.

18 Sabrina Jaeger, Karsten Klein, Lucas Joos, Johannes Zagermann, Michael de Ridder, Jinman Kim, Jean Yang, Ulrike Pfeil, Harald Reiterer, and Falk Schreiber. Challenges for brain data analysis in VR environments. In *2019 IEEE Pacific Visualization Symposium (PacificVis)*, pages 42–46, 2019. doi:10.1109/PacificVis.2019.00013.

19 Lucas Joos, Maximilian T. Fischer, Daniel A. Keim, and Johannes Fuchs. Aesthetic-driven navigation for node-link diagrams in vr. In *Proceedings of the 2023 ACM Symposium on Spatial User Interaction*, SUI '23. ACM, 2023. doi:10.1145/3607822.3614537.

20 Lucas Joos, Maximilian T. Fischer, Julius Rauscher, Daniel A. Keim, Tim Dwyer, Falk Schreiber, and Karsten Klein. Visual network analysis in immersive environments: A survey, 2025. doi:10.48550/arXiv.2501.08500.

21 Lucas Joos, Sabrina Jaeger-Honz, Falk Schreiber, Daniel A. Keim, and Karsten Klein. Visual comparison of networks in vr. *IEEE Transactions on Visualization and Computer Graphics*, 28(11):3651–3661, 2022. doi:10.1109/TVCG.2022.3203001.

22 Lucas Joos, Gavin J. Mooney, Maximilian T. Fischer, Daniel A. Keim, Falk Schreiber, Helen C. Purchase, and Karsten Klein. Replication Data for: "Show Me Your Best Side: Characteristics of User-Preferred Perspectives for 3D Graph Drawings", 2025. doi:10.18419/DARUS-5108.

23 Steve Kieffer, Tim Dwyer, Kim Marriott, and Michael Wybrow. HOLA: Human-like orthogonal network layout. *IEEE Transactions on Visualization & Computer Graphics*, 22(01):349–358, 2016. doi:10.1109/TVCG.2015.2467451.

24 Roman Klapaukh, Stuart Marshall, and David Pearce. A symmetry metric for graphs and line diagrams. In *Diagrammatic Representation and Inference*, pages 739–742. Springer International Publishing, 2018. doi:10.1007/978-3-319-91376-6_71.

25 Maxim Kuznetsov, Aviv Elor, Sri Kurniawan, Colleen Bosworth, Yohei Rosen, Nicholas Heyer, Mircea Teodorescu, Benedict Paten, and David Haussler. The immersive graph genome explorer: Navigating genomics in immersive virtual reality. In *2021 IEEE 9th International Conference on Serious Games and Applications for Health(SeGAH)*, pages 1–8, 2021. doi:10.1109/SEGAH52098.2021.9551857.

26 Michael R. Marner, Ross T. Smith, Bruce H. Thomas, Karsten Klein, Peter Eades, and Seok-Hee Hong. GION: Interactively untangling large graphs on wall-sized displays. In *Graph Drawing: 22nd International Symposium*, volume 8871 of *LNCS*, pages 113–124. Springer, 2014. doi:10.1007/978-3-662-45803-7_10.

27 Fintan McGee, Mohammad Ghoniem, Guy Melançon, Benoit Otjacques, and Bruno Pinaud. The state of the art in multilayer network visualization. *Computer Graphics Forum*, 38(6):125–149, 2019. doi:10.1111/cgf.13610.

28 Michael J. McGuffin, Ryan Servera, and Marie Forest. Path tracing in 2d, 3d, and physicalized networks. *IEEE TVCG*, 30(7):3564–3577, 2024. doi:10.1109/TVCG.2023.3238989.

29 Amyra Meidiana, Seok-Hee Hong, Peter Eades, and Daniel Keim. Quality metrics for symmetric graph drawings. In *2020 IEEE Pacific Visualization Symposium (PacificVis)*, pages 11–15, 2020. doi:10.1109/PacificVis48177.2020.1022.

30 Gavin J. Mooney, Helen C. Purchase, Michael Wybrow, and Stephen G. Kobourov. The multi-dimensional landscape of graph drawing metrics. In *2024 IEEE 17th Pacific Visualization Conference (PacificVis)*, pages 122–131, 2024. doi:10.1109/PacificVis60374.2024.00022.

31 Gavin J. Mooney, Helen C. Purchase, Michael Wybrow, Stephen G. Kobourov, and Jacob Miller. The perception of stress in graph drawings. In *32nd International Symposium on Graph Drawing and Network Visualization (GD 2024)*, volume 320 of *LIPIcs*, pages 21:1–21:17. Schloss Dagstuhl, 2024. doi:10.4230/LIPIcs.GD.2024.21.

32 Martin Nöllenburg, Sebastian Röder, and Markus Wallinger. GdMetriX - A NetworkX extension for graph drawing metrics. In *32nd International Symposium on Graph Drawing and Network Visualization (GD 2024)*, volume 320 of *LIPIcs*, pages 45:1–45:3. Schloss Dagstuhl, 2024. doi:10.4230/LIPIcs.GD.2024.45.

33 Rusheng Pan, Helen C. Purchase, Tim Dwyer, and Wei Chen. Extending adjacency matrices to 3d with triangles. In *2024 IEEE 17th Pacific Visualization Conference (PacificVis)*, pages 232–241, 2024. doi:10.1109/PacificVis60374.2024.00033.

34 Sebastian Pirch, Felix Müller, Eugenia Iofinova, Julia Pazmandi, Christiane V. R. Hütter, Martin Chiettini, Celine Sin, Kaan Boztug, Iana Podkosova, Hannes Kaufmann, and Jörg Menche. The vrnetzer platform enables interactive network analysis in virtual reality. *Nature Communications*, 12(1):2432, 2021. doi:10.1038/s41467-021-22570-w.

35 Helen C. Purchase. Metrics for graph drawing aesthetics. *Journal of Visual Languages & Computing*, 13(5):501–516, 2002. doi:10.1006/jvlc.2002.0232.

36 Helen C. Purchase, Christopher Pilcher, and Beryl Plimmer. Graph drawing aesthetics–created by users, not algorithms. *IEEE Transactions on Visualization & Computer Graphics*, 18:81–92, December 2012. doi:10.1109/TVCG.2010.269.

37 Kathy Ryall, Joe Marks, and Stuart Shieber. An interactive constraint-based system for drawing graphs. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology*, UIST '97, pages 97–104. ACM, 1997. doi:10.1145/263407.263521.

38 Alexander Schaefer, Ru Kong, Evan M. Gordon, Timothy O. Laumann, Xi-Nian Zuo, Avram J. Holmes, Simon B. Eickhoff, and B. T. Thomas Yeo. Local-global parcellation of the human cerebral cortex from intrinsic functional connectivity mri. *Cerebral Cortex*, 28(9):3095–3114, 2017. doi:10.1093/cercor/bhx179.

39 Miro Spönemann, Björn Duderstadt, and Reinhard von Hanxleden. Evolutionary meta layout of graphs. In *Diagrammatic Representation and Inference*, pages 16–30. Springer, 2014. doi:10.1007/978-3-662-44043-8_3.

40 Martyn Taylor and Peter Rodgers. Applying graphical design techniques to graph visualisation. In *Ninth International Conference on Information Visualisation (IV'05)*, pages 651–656, 2005. doi:10.1109/IV.2005.19.

41 Floris Tijhuis, Minne Schepers, Eduarda Centeno, Bernardo de Maciel, Linda Douw, and Fernando Nobrega Santos. Human connectome project rfmri repository. https://github.com/floristijhuis/HCP-rfMRI-repository, 2025. San Francisco (CA): GitHub; [Last Accessed 25.07.2025].

42 Floris Benjamin Tijhuis, Minne Schepers, Eduarda Centeno, Bernardo de A.P.C. Maciel, Linda Douw, and Fernando Nobrega Santos. Human connectome project resting-state fmri connectivity matrices (young adult + aging), June 2022. Zenodo. doi:10.5281/zenodo.6770120.

43 Simon van Wageningen, Tamara Mchedlidze, and Alexandru Telea. An experimental evaluation of viewpoint-based 3D graph drawing. *Computer Graphics Forum*, 43(3), 2024. doi:10.1111/cgf.15077.

44 Simon van Wageningen, Tamara Mchedlidze, and Alexandru Telea. Viewpoint Optimization for 3D Graph Drawings. *Computer Graphics Forum*, 44(3), 2025. doi:10.1111/cgf.70127.

45 Colin Ware, Helen Purchase, Linda Colpoys, and Matthew McGill. Cognitive measurements of graph aesthetics. *Information Visualization*, 1(2):103–110, 2002. doi:10.1057/palgrave.ivs.9500013.

46 Eric W. Weisstein. Circle-circle intersection. https://mathworld.wolfram.com/Circle-CircleIntersection.html. From MathWorld–A Wolfram Web Resource. [Last accessed 25.07.2025].

47 Eric Welch and Stephen Kobourov. Measuring symmetry in drawings of graphs. *Computer Graphics Forum*, 36(3):341–351, 2017. doi:10.1111/cgf.13192.

48 Vahan Yoghourdjian, Yalong Yang, Tim Dwyer, Lee Lawrence, Michael Wybrow, and Kim Marriott. Scalability of network visualisation from a cognitive load perspective. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1677–1687, 2021. doi:10.1109/TVCG.2020.3030459.

49 Ying Zhang, Niklas Groene, Karsten Klein, Giuseppe Liotta, and Falk Schreiber. Investigating crossing perception in 3d graph visualisation, 2025. doi:10.48550/arXiv.2508.00950.

Stress in Graph Drawings: Perception, Preference, and Performance

Gavin J. Mooney
Monash University, Melbourne, Australia

Jacob Miller
Technical University of Munich, Heilbronn, Germany

Michael Wybrow
Monash University, Melbourne, Australia

Stephen Kobourov
Technical University of Munich, Heilbronn, Germany

Helen C. Purchase
Monash University, Melbourne, Australia

Abstract

Stress in a graph drawing has been a popular layout principle for more than two decades. Low stress drawings exhibit the property that the geometric distances between all pairs of nodes correlate with the shortest paths between them. The assumption has always been that low stress drawings are "nicer" and better support human perception and comprehension than high stress drawings. In this paper, we put these assumptions to the test. We use a normalised scale-independent and rotation-independent metric for stress; this is necessary to ensure strict controls on our experimental stimuli. We report on three experiments, exploring human perception of stress, preference for stress, and the effect of stress on a graph performance task. We conclude that people can see stress in a graph drawing, that they prefer low stress drawings, and that their performance in a shortest path task improves as stress decreases – thus empirically confirming long-standing assumptions.

2012 ACM Subject Classification Human-centered computing → Graph drawings

Keywords and phrases Graph Drawing, Graph Drawing Metrics, Stress, Visual Perception, User Study

Digital Object Identifier 10.4230/LIPIcs.GD.2025.38

1 Introduction

Since its introduction in the 1980s [12], minimising stress has remained one of the most popular layout criteria in graph drawing with implementations in GraphViz [10] and NetworkX [19]. It is among the most frequent quality metrics that researchers use to evaluate layout algorithms, being the third most used metric behind run time and edge crossings [7]. Although stress is believed among experts to reflect aesthetic, symmetric, and general high quality layouts [27], validating stress as a metric has only been considered in a handful of limited studies [5, 18]. The assumption is that humans can perceive stress, prefer low stress, and understand low stress drawings better than high stress ones. This paper reports on empirical studies that address these assumptions.

While the fundamental principle of measuring stress relates to the extent to which the geometric distances between pairs of nodes correlate with the graph theoretic distances between them, there are several different definitions of stress. Many formulae for stress depend on the scale of the drawing [24] – that is, since the physical size of a drawing determines geometric distances, it also determines the value of the drawing's stress. These


© Gavin J. Mooney, Jacob Miller, Michael Wybrow, Stephen Kobourov, and Helen C. Purchase;
licensed under Creative Commons License CC-BY 4.0



ambiguities in the definition of stress means that it is difficult to compare the stress of two graph drawings of different size, and not easy to create a unique graph drawing with given stress.

To investigate the human response to varying stress in graph drawings, we use a scale-independent and rotation-independent normalised metric for stress [18]. Having a robust metric like this enables rigorous formal experiments to be conducted, since it provides a means by which graph drawings of different size and scale can be quantitively compared for the extent of stress within them.

Based on this stress metric, a large set of graph drawings with different stress values served as stimuli for our three human empirical studies. The first experiment addresses the question of whether people can see stress in a graph drawing or not, gathering qualitative interview data that validates the results of prior quantitative work [18]. We find that while participants can recognise differences in stress, they have difficulty describing what it is. Since these "perception" results may have been due to the preferences of the participants for low stress drawings, our second experiment investigated preference, asking participants to identify which of a several pairs of drawings they most preferred in a two-alternative-forced-choice experiment. We found that they did, indeed, prefer the drawing with lower stress, but that this was not the same as comparing two drawings and identifying the one with least stress. The final experiment is arguably the most important, since it focusses on actual task performance, asking participants to complete shortest path tasks. As we had hoped, the results reveal a correlation between task accuracy and stress: the lower the stress, the higher the accuracy.

We have therefore considered stress in graph drawings from three human perspectives: perception, preference and performance.

2 Background: Stress in Graph Drawings

The concept of stress in graph drawing is based on the principle that any two vertices should have a geometric distance proportional to their graph theoretic distance. While there are several variants for the definition of stress, the one proposed by Gansner et al. [10] is most popular (Equation 1).

$$\sum_{i<j} d_{i,j}^{-2}(||X_i - X_j|| - d_{i,j})^2 \tag{1}$$

where X_i is the coordinate position of vertex i and $d_{i,j}$ is the graph theoretic distance between vertices i and j. Gansner et al. [10] note that the sum of square differences over-penalises large differences and under-penalises small distances. Hence, the inclusion of the $d_{i,j}^{-2}$ "normalisation" term, weighting distant pairs proportionally to an inverse-square law; thus this definition is known as *normalised stress*. This equation has roots in statistical analysis and dimension reduction, where it is known as Multi-dimensional Scaling (MDS) or Sammon mapping [14, 22].

Stress was first introduced to graph drawing as an objective function, that is, something to be minimised to produce a desirable drawing, and several variants of Equation 1 have been proposed as objective functions in various algorithms. Gansner et al. [9] and Ortmann et al. [20] use approximation in order to avoid the expensive all-pairs-shortest-path computation that is required for normalised stress. Chen and Buja [4] and Miller et al. [16] modify the function to measure only "local" (small) distances.

When optimising stress, Ahmed et al. [1] and Devkota et al. [6] use stress-based optimisation schemes alongside other aesthetic criteria, and stress is used as an evaluation metric by Hong et al. [11] and Marmer et al. [15], as well in dynamic graph layout methods from Simonetto et al. [23] and Arleo et al. [2]. Kruiger et al. [13] and Zhu et al. [29] evaluate the stress of their layouts to determine how much "global" distances are lost.

When investigating stress, Brandes and Pich [3] evaluate layout algorithms by stress and Welch and Kobourov [27] use stress as an alternative measure of symmetry in graph layouts. Wang et al. [26] both optimise and evaluate stress. Wageningen et al. [25] extend stress as an evaluation for 3-dimensional drawings, by considering each orthographic viewpoint.

2.1 Scale-Independent Stress

Normalised stress (Equation 1) is susceptible to changes in the scale (size) of a drawing [24]: that is the same drawing will have different stress values when drawn at different sizes. While this may not be a concern when the aim of a layout algorithm is to minimise the stress in a drawing, it means that the measured stress value of a drawing is objectively meaningless outside of the specific implementation. In particular, it means that it is impossible to meaningfully compare the stress vales of two graph drawings.

If we are to empirically investigate the effect of stress on human perception, preference and performance, we need a stress metric that is scale-invariant and bounded. There are three main contenders. Scale-normalised stress has been deployed in recent evaluations [25, 26], where the size of the drawing is chosen which yields the smallest normalised stress, and the Shepard goodness score measures the rank correlation between the graph-theoretic and geometric distances between vertices.

The option we have chosen is the non-metric stress of Kruskal [14], which has several desirable properties as a metric. Unlike normalised stress which is unbounded and scale-sensitive, non-metric stress lies in the range $[0, 1]$ and is scale-invariant. The two stress functions share the same overall complexity to compute (quadratic, once all-pairs shortest-path lengths are precomputed), and the difference in runtime is also small in practice. Mooney et al. [18] show that this Kruskal's stress metric (KSM) is highly correlated with Gansner et al.'s [10] normalised stress.

$$KSM = 1 - \sqrt{\frac{\sum_{i<j}(||X_i - X_j|| - \hat{d}_{i,j})^2}{\sum_{i<j} ||X_i - X_j||^2}} \tag{2}$$

where X_i is the position of vertex i, and $\hat{d}_{i,j}$ is the horizontal distance to the monotonic regression line of best fit in the Shepard diagram of the drawing. A more detailed description is given in [24]. Note that, consistent with other literature on normalised evaluation metrics (e.g., [17,18,21]), a value of 1 represents the assumed "best" case (i.e., zero stress). Therefore, high KSM values indicate low stress.

There are few empirical studies that compare the effectiveness of stress. Computational experiments have shown that the low stress of a drawing tends to correspond to exhibiting symmetries in the graph when they exist [27]. A small-scale experiment indicates that people prefer less stress and fewer crossings [5]. However, in this experiment, it is unclear the level of difference of stress values between drawings, or even what variant of stress was deployed (stated "scaled by average edge length"). The measure of stress used relies on normalising by average edge lengths, which may invalidate comparisons between drawings where the average edge length (or scale of the overall drawing) are different. Our use of the scale-invariant and bounded KSM allows for meaningful stress comparison between graph drawings, ensuring that valid experimental stimuli can be created.

Other studies consider the length of shortest paths and Euclidean distances without referring to stress directly. Yoghourdjian et al. [28] evaluate shortest path task difficulty with respect to features of the shortest path, including a stress variant called "geodesic path deviation". However, they find that this feature of the shortest path does not influence task difficulty as much as several other layout parameters that relate to crossings (e.g. number of edge crossings, angle of crossings, node-edge crossings).

2.2 Experimental Stimuli

Our experiments require a set of graph drawings with known stress values; a basic hill climbing algorithm was used to optimise the Kruskal Stress Metric (KSM) described in Section 2.1. Drawings were generated with distinct KSM values from 0.4 to 0.8, incrementing by 0.05. This gives nine unique stress values and nine unique differences in stress values between pairs of drawings, which we refer to as "deltas". For example, for a pair of drawings with KSM of 0.45 and 0.55, the delta would be 0.1, thus the deltas range from 0 to 0.4. In total, 405 graph drawings were created, from 15 randomly generated graphs at three sizes (10, 25, 50 nodes). Mooney et al. [18] describe the process for creating these drawings in more detail. It is important to note that KSM assigns a value of 1 to represent the ideal case (zero stress); thus "low stress" has a high KSM value; "high stress" has a low KSM value.

This set of drawings is used in all three experiments described below. Example stimuli are shown in Figure 1.

3 Stress Perception Experiment

In this section we present results of an extended version of an experiment that investigated whether humans can perceive stress in a graph drawing [18]. The stress perception experiment validates and extends the work of Mooney et al. [18], with the inclusion of more participants and some interviews.

3.1 Methodology

Three cohorts – trained novices (25), untrained novices (25), and experts (9) – were asked to determine which of a pair of graph drawings had the lowest stress (or if they had the same stress) in an online survey offered through the "Qualtrics" platform. Three graph sizes were shown to participants, resulting in a total of 177 participant data points. Accuracy, response time, and confidence data were collected. Participants were also asked to describe the strategies that they used to determine stress in the drawings. In-person interviews with five novice participants aimed to gain a deeper understanding of participants' strategies when comparing stress. Interview participants were also asked to explain what "stress" is; we wanted to see if participants really understood the theoretical notion of "stress". The methodology is otherwise the same as Mooney et al. [18].

3.2 Summary of results

The results reinforced those of Mooney et al. [18], indicating that even untrained novices can identify stress, although less successfully than trained novices or experts. Surprisingly, the size of the graph (number of nodes) did not alter these results. Participants devised (visible) proxies for (invisible) stress – for example, the length of edges, the distances between nodes, extent of edge crossings, and node distribution, using words like "compactness", "clustering",

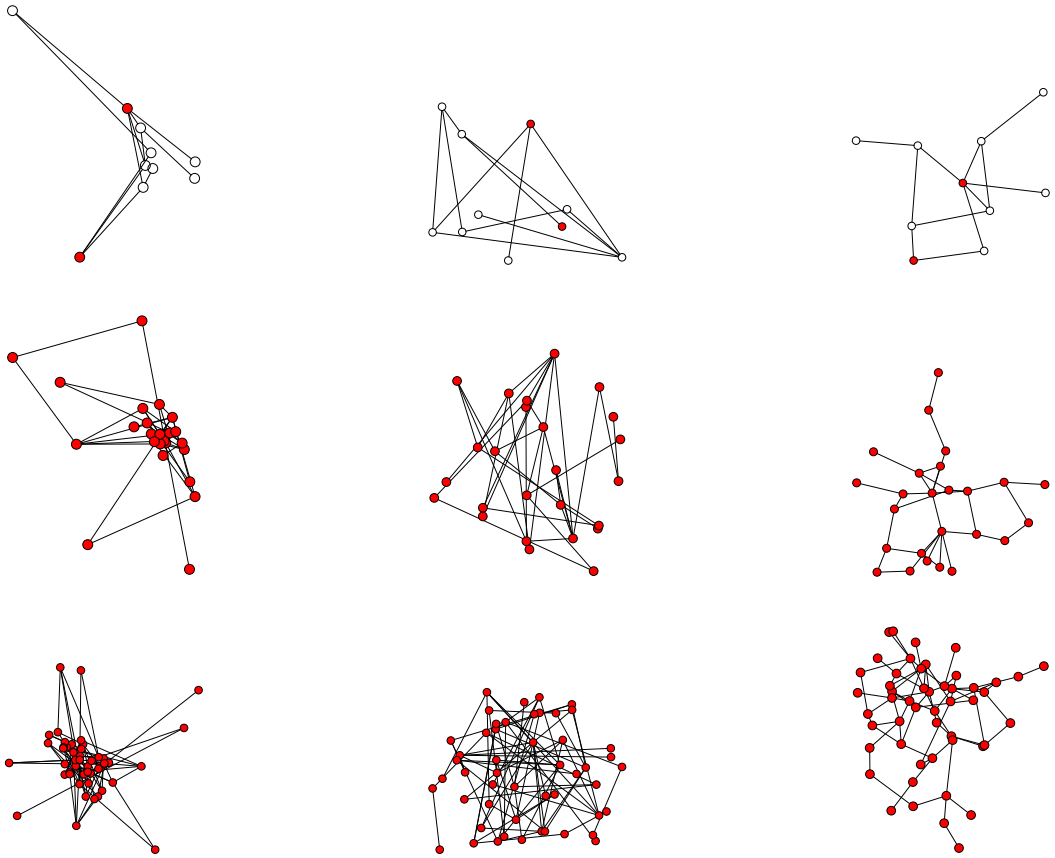


Figure 1 Example stimuli used in the experiments. The top row shows three drawings of the same 10-node graph with different highlighted end points for shortest paths at KSM values of 0.4, 0.6, and 0.8 (left to right), used in the *Shortest Path Experiment*. The middle and bottom rows show drawings of the same 25-node and 50-node graphs, respectively, at KSM values of 0.4, 0.6, and 0.8 (left to right), used in the *Perception* and *Preference* experiments.

"density" – we call these *"Stress Identification Proxies"*. Only one of the interview participants gave a reasonable definition of stress ("it's the way the objects are positioned that matches the length of paths."). We conclude that while it is unclear whether participants utilised the exact geometric definition of stress their performance suggests that they were able to perceive it ("I know it when I see it").

4 Stress Preference Experiment

While the results of the Stress Perception Experiment in the previous section suggest that participants can correctly identify the difference between high and low stress drawings, it is not clear that they can "see" stress in the literal sense. Perhaps they simply chose the drawing that they preferred?

In this section, we describe a two-alternative forced choice experiment that simply asks participants to choose which of two graph drawings they prefer. We use the same stimuli from Mooney et al. [18]: 405 graph drawings with KSM stress values ranging from 0.4 to 0.8, in 0.05 increments (Section 2.2), using graphs with 10, 25 and 50 nodes and 25 participants per graph size.

4.1 Methodology

Participants were shown 45 pairs of drawings and asked to choose their preferred drawing (left or right). We represent these choices as a binary value: 1, if the participant selected the drawing with lower stress (to match the accuracy measure for the stress perception experiment), and 0 if they chose the drawing with higher stress. To be consistent with the prior perception experiment, the trials where stress is the same in both drawings were shown to participants but the responses discarded from analysis.

Participants in these experiments were given a layperson's description of networks, but no specific information on paths or stress. At the end of the experiment they were asked demographic questions (including their familiarity with "stress in a graph drawing") and asked "Please describe in your own words which visual aspects of the drawings affected your choice."

4.2 Results

75 people participated in the Stress Preference Experiment (25 per graph size). Figure 2 shows the distribution of preferences for each participant, grouped by graph size. If stress did not affect preference choice, we would expect these distributions to approximate normal distributions centred around a mean of 50%, since each data point is the percentage of times a participant chose the lower stress drawing, and 50% represents a random choice. The high median values (>80%) and skewed distributions show a tendency for preference of lower stress drawings. A Wilcoxon signed-rank test for each graph size (comparing observed distributions against the median of 50% expected for a random choice) gave p-values <0.001 for all graph sizes. Thus, each distribution in Figure 2 differs significantly from random chance and we conclude stress affects preference choices.

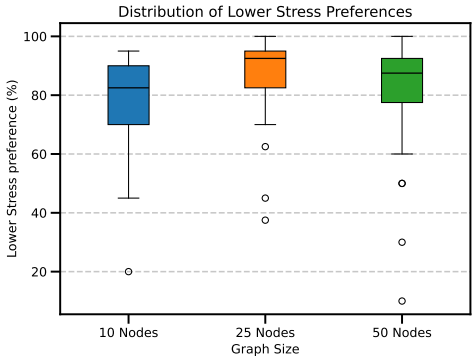


Figure 2 Box plot showing the distribution of preferences for each participant. Each data point represents the percentage of times the participant preferred the lower stress drawing. Skewed distributions show a preference for low stress drawings for all three graph sizes (10, 25, 50).

Figure 3 presents a heatmap showing the percentage of times each individual drawing was preferred, in relation to the number of times it was shown: the "preference ratio". This shows that low stress (high KSM) drawings were chosen more frequently than high stress (low KSM) drawings. The outliers can be explained by the fact that drawings were chosen randomly and so some were only shown a few times, and with small differences in KSM values between the pairs.

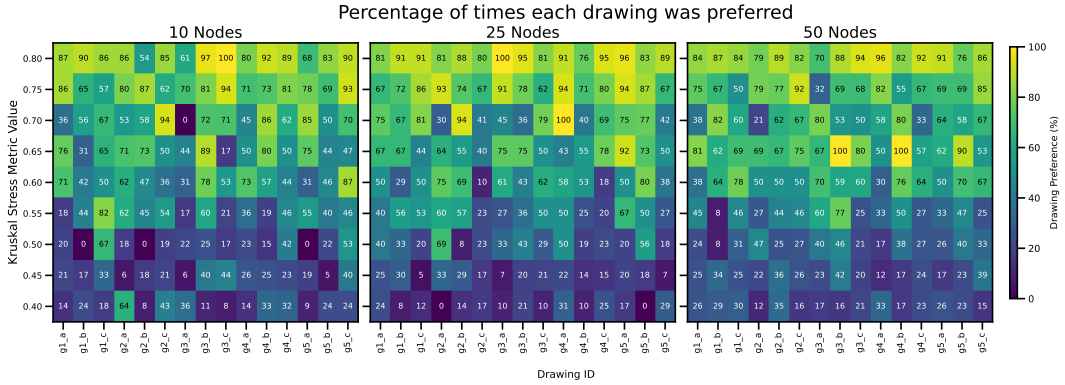


Figure 3 Heatmap showing the percentage of times each drawing was selected as the preferred choice, in relation to the number of times shown.

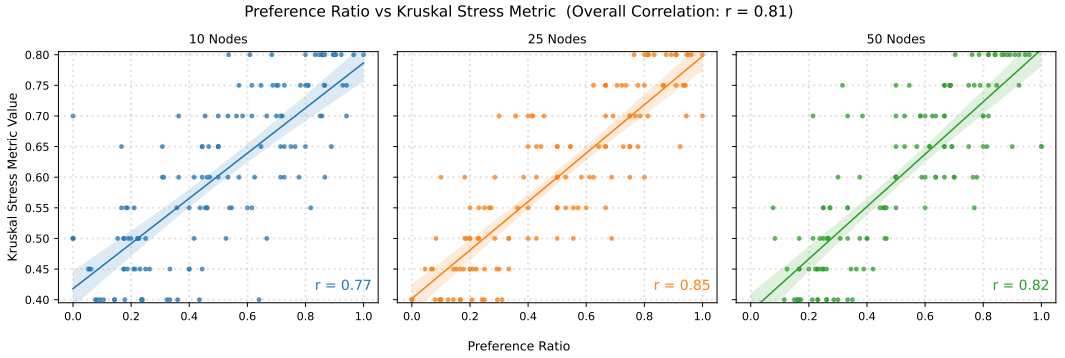


Figure 4 The Preference ratio of each drawing plotted against its KSM value. The linear regression lines have confidence intervals shown by shading. The trends show a preference for low stress (high KSM) drawings across all three graph sizes (10, 25, 50).

We plot the KSM values against the preference ratios over all graph sizes in Figure 4, revealing that the preference ratio of a drawing tends to increase as KSM increases (regardless of graph size). The Pearson correlation between preference and KSM (over all graph sizes) is 0.81.

Participants mentioned factors influencing their preference, including: more space between nodes (39), overall shape/symmetry (7), space between edges (9), fewer edge crossings (21), and less clutter/mess (10). Some responses can be interpreted as direct references to existing graph drawing metrics: "I disliked overlapping edges, and preferred evenly spaced nodes." (Edge Crossings, Node Uniformity); while others were less specific: "How messy it appeared to my brain". One participant mentioned multidimensional scaling directly: "I found myself preferring the pattern that looks more multidimensional. Multidimensional scaling appears more structured and provides more information about the connections between the nodes and the distances among them."

From the participants' responses, we identify five *Stress Preference Proxies*: Angular Resolution, Edge Lengths, Node Uniformity, Edge Crossings, and Gabriel Ratio. The correlation between these five proxies (measured with metrics defined by Mooney et al. [17]) and the KSM stress metric over all 405 stimuli are: Angular Resolution (0.61), Edge Lengths (0.78), Node Uniformity (0.69), Edge crossings (0.66) and Gabriel Ratio (0.84), suggesting

that people prefer low stress drawings. Although their decision is based on identifiable visual proxies, underlying these proxies is the unseen layout principle of "stress". The distributions of metric values for all 405 drawings in our dataset are in Appendix B.

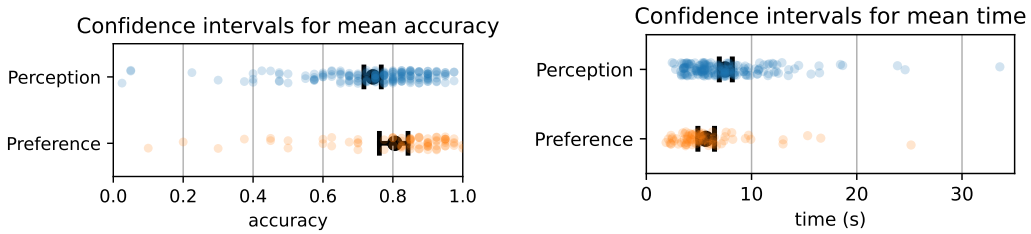
4.3 Perception vs Preference

One of our motivations for following the Stress Perception Experiment with the Stress Preference Experiment was to determine whether our results of the former were simply based on participants' preference for lower stress drawings (rather than on perception of stress). To investigate this, we compare the 177 data points from the Perception Experiment with the 75 from the Preference Experiment.

4.3.1 Results

Figure 5 (a) shows the distribution of mean accuracy across the aggregated groups in both experiments, with bootstrapped 95% confidence intervals. We compare the distributions for similarity (excluding same-stress pairs). Note that while we use the term "accuracy' there are no objectively "correct' answers in the preference experiment. Here, accuracy refers to whether the participant selected the drawing with lower stress.

A Wilcoxon rank sum test between the Perception data and the Preference data gives a p-value of 0.0189 (less than 0.05) indicating a significant difference, albeit with a small effect size. Thus, getting the correct answer in the Perception experiment is not the same as preferring lower stress.



(a) (b)

Figure 5 Bootstrapped 95% confidence intervals for (a) mean accuracy and (b) mean response time in the Perception and Preference experiments, aggregated over all graph sizes.

Figure 5 (b) shows the distribution of mean response time across the aggregated groups in both experiments, with bootstrapped 95% confidence intervals. We compare the distributions for similarity (excluding same-stress pairs). A Wilcoxon rank sum test between the Perception data and the Preference data gives a p-value of 2.6×10^{-6} (less than 0.05) indicating a significant difference. Thus, the cognitive effort required for selecting the preferred drawing (regardless of accuracy) was less than that required for comparing stress.

4.3.2 Discussion

From the Perception Experiment, we conclude that participants can 'see' stress, and can identify the drawing with less stress within a pair (even if they can't explain what 'stress' actually means). From the Preference Experiment, it is clear that participants prefer drawings with lower stress. Our comparative analysis suggests, however, that when asked to choose the drawing with least stress (in the earlier experiment), participants were not simply choosing

the one they most preferred – they really were considering their perception of stress. This is shown both by the differences in the choice of the least stress drawing (Figure 5 (a)) as well in the difference in the time taken to make the decision (Figure 5 (b)). The difference in response time shows that the perception task took longer than the preference task. This result suggests that the task in the Perception Experiment had higher cognitive load (with participants thinking carefully about stress in terms of graph structure and relative distances) while the Preference task was enacted more intuitively, possibly based on immediate perception rather than cognitive analysis.

We note that, while the difference in means between the two experiments is quite small, it is still statistically significant. While we conclude that perceiving stress is distinct from preferring lower stress, one is certainly a very strong predictor of the other.

5 Stress Performance

Assessing whether humans can see stress and what stress levels they prefer is all very well, but task performance is what really matters. If we are to use graphs to depict information in a meaningful way, so that it can be accurately used and understood, then stress perception and preference is insufficient: we need to know how stress affects human"s ability to read the information embodied in the graph. This Stress Performance experiment uses a shortest path task to measure participant"s ability to interpret the graph.

5.1 Methodology

Participants in this study, conducted via the new reVISit framework [8], were shown a series of drawings of graphs, which had two vertices highlighted in red – all other vertices were transparent to reduce uncertainty when following edges which intersect nodes. They were then asked, "What is the shortest path between the two highlighted nodes?" The possible answers available were 2, 3, 4 or 5. Participants were also given the option "I am unable to work this out".

The stimuli were sourced from the perception and preference experiments described earlier. We use a subset of drawings (243 of the original 405) to ensure an equal number of responses for each drawing. The highlighted nodes were based on shortest paths randomly selected through repeated sampling to ensure an even distribution of path lengths (2, 3, 4, or 5) across KSM values and graph sizes. Due to the structure of the 10-node graphs, their corresponding stimuli only included shortest paths up to length 4.

Participants were given a short description about graphs, graph drawings, and shortest paths. These descriptions are made available through supplemental material (Appendix A). Participants were then informed they would be shown 6 training examples, that we discard from analysis. These examples provided feedback on whether the participant answered the task correctly, and were the same for each participant.

We used a within-subjects study design, where each participant sees all conditions during the study. In this case, each participant saw one example of each stress level (9) for every size of graph (3). Each size was broken into its own block during the study, so that for a given size, all trials are shown sequentially. After seeing all trials of a size, participants could take a short break. Additionally, each size block included two additional fixed trials at the beginning which were discarded from the analysis to account for learning effects. In total, this amounts to $(9 + 2) \times 3 = 33$ trials per participant, after the training.

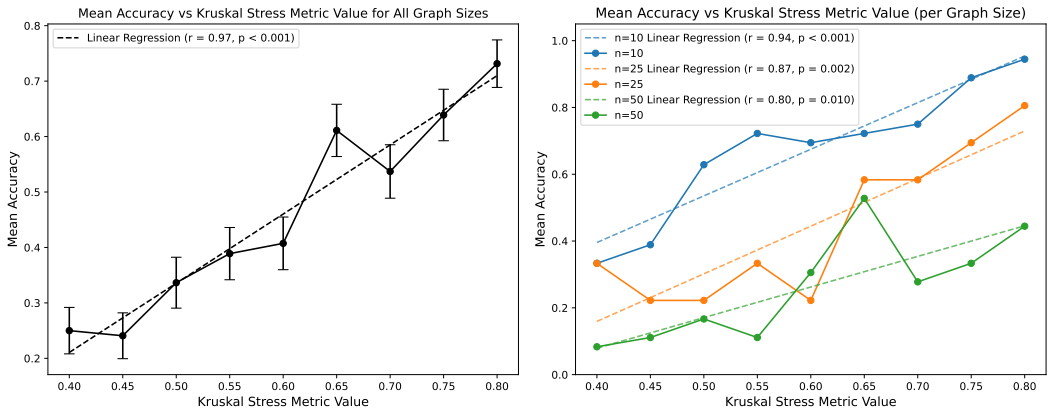
The exact stimuli used during the study varied per participant. We partitioned the drawings used from the earlier two studies into nine disjoint groups, and assigned participants a group number based on a Latin square, ensuring the same number of participants in each

group. The order of trials was randomised in two ways: first the size blocks were shown in random order, and second within each block the order of the stimuli was randomised. Together these aimed to reduce learning effect noise in the collected data. Finally, after all 33 trials, participants were asked to provide basic demographic information the same as in the previous two experiments.

5.2 Results & Discussion

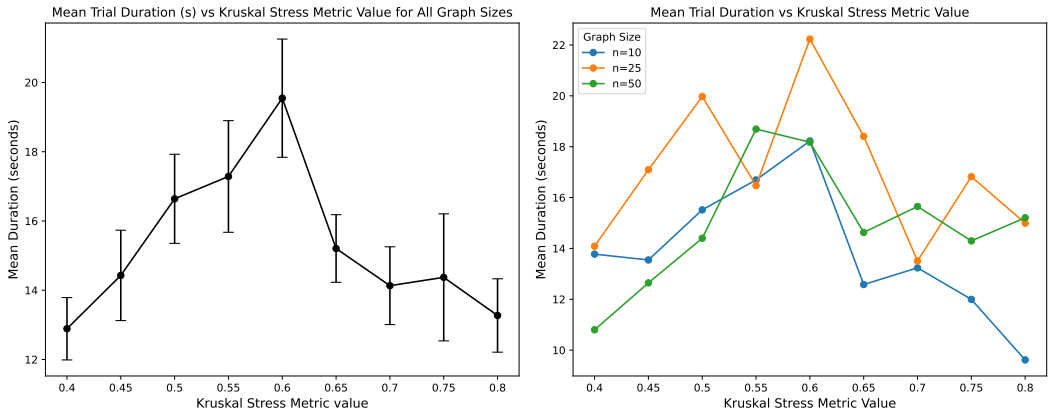
Figure 6 (a) plots the mean accuracy against the KSM values over all graph sizes, showing a clear upward slope: accuracy increases as stress decreases (where high KSM indicates low stress). The linear fit line indicates a correlation of 0.97 with $p < 0.001$. Figure 6 (b) shows the results according to graph size: clearly the larger the graph, the harder the task. The lower correlation value for n=50 (0.80) compared with n=10 (0.94) suggests that the relative effect of stress on performance is diminished for the larger graphs. From this, we might even anticipate that for very large graphs, say n=1000, stress has no effect on task performance; this is an interesting avenue for future work.

Figure 7 (a) shows the mean duration of trials for each KSM value. We suggest the following explanations for the shape of this trend: Drawings with high KSM (low stress) made the task easier, and thus participants required less time to work out the answer. Drawings with low KSM (high stress) were more difficult, causing participants to realise quickly that they cannot easily find the correct answer. The middle KSM values indicate longer response times where the drawing made the task neither too easy nor too difficult. This is also supported by Figure 7 (b), which shows that the trials for n=25 had the longest duration (n=10: too easy; n=25: neither too easy nor too difficult; n=50: too difficult). Figure 8 (a) further supports this argument, showing that participants were unable to determine the answer for low KSM drawings. Figure 8 (b) shows the number of abandoned trials for each graph size, further highlighting the difficulty of the task on larger graphs.



(a) (b)

Figure 6 Relationship between mean accuracy and Kruskal stress metric values. (a) Aggregated across all graph sizes. (b) Disaggregated by individual graph sizes. Error bars represent standard error of the mean. Linear regression lines indicate a positive correlation between stress and accuracy for all graph sizes ($r = 0.97$, $p < 0.001$), with strongest correlation observed for smaller graphs ($n = 10$: $r = 0.94$, $p < 0.001$; $n = 25$: $r = 0.87$, $p = 0.002$; $n = 50$: $r = 0.80$, $p = 0.01$).



(a) (b)

Figure 7 Relationship between mean task duration (in seconds) and Kruskal stress metric values. (a) Aggregated across all graph sizes. (b) Disaggregated by individual graph sizes. Error bars represent standard error of the mean.

5.2.1 Anomalies

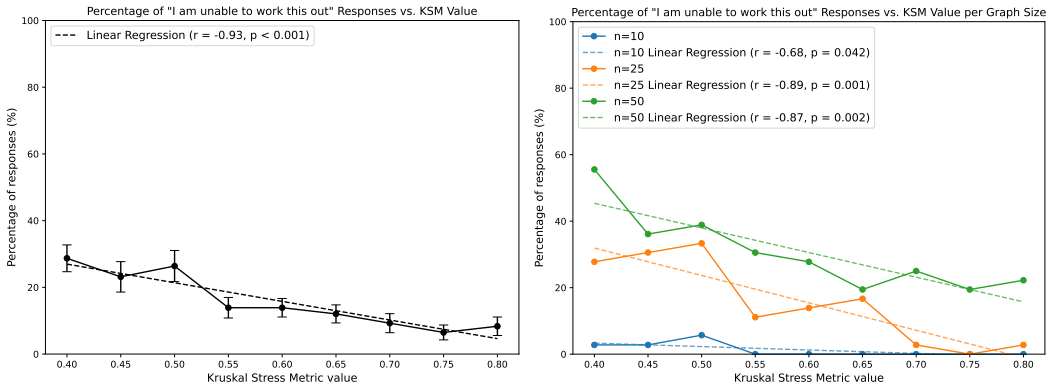
There are three noticeable anomalies in Figure 6 (b):

- For n=25, KSM=0.60, the task seems to have been unusually difficult;
- For n=50, KSM=0.60, the task seems to have been unusually easy;
- For n=50, KSM=0.65, the task seems to have been unusually easy.

We investigated these anomalies by analysing the stimuli, considering the accuracy for individual drawings (Figure 9) and features of the drawings that may have made them particularly easy or difficult. For example, we looked at whether the shortest path was unique, whether either the highlighted source or target nodes were of degree 1, or if they overlapped another node, or whether an edge on the path was shorter than the radius of a node (and therefore shown within the intersection of two nodes). Most of these features were revealed to have little effect (see the heatmaps in Appendix C), but some of the data helps explain our three anomalies:

- Eight out of the nine n=25, KSM=0.60 drawings had edges on the shortest path that were directly on top of another edge (more than any other set of n=25 graphs when grouped by KSM). In addition, five of the nine n=25, KSM=0.60 drawings had a source or target node overlapping an unconnected edge. The combination of these two factors would have made these drawings naturally more difficult than the other n=25 ones.
- Fewer n=50, KSM=0.60 and n=50, KSM=0.65 drawings (three and four respectively) had source or target nodes overlapping another node (as opposed to five or more for other n=50 drawings grouped by KSM); there were similarly fewer drawings where a node on the shortest path overlaps another node. The combination of these two factors would have made these drawings naturally easier than the other n=50 ones.

While we focused on the three obvious anomalies in Figure 6 (b), our investigation revealed other sets of drawings that may have been easier than others of the same size. Having source or target nodes on the convex hull might make the task easier; this was the case for drawings of size 25 with KSM=0.55. Similarly, a shorter average geodesic node distance (average of the perpendicular distances between nodes on the shortest path and a straight line between the source and target) might also simplify a shortest path task; this was the case for both n=25, KSM=0.55 and n=50, KSM=0.5 drawings. These smaller variations can also be seen in the results chart in Figure 6 (a).



(a) (b)

Figure 8 Relationship between the percentage of "I am unable to work this out" responses and Kruskal stress metric values. (a) Aggregated across all graph sizes. (b) Disaggregated by individual graph sizes. Error bars represent standard error of the mean. Linear regression lines indicate a negative correlation between Kruskal stress and task abandonment for all graph sizes ($r = -0.93$, $p < 0.001$).

In exploring these anomalies, one issue became increasingly apparent: *an alteration in stress in a drawing naturally affects other visual features.* If a graph drawing is generated by an algorithm that only cares about the amount of stress, then undesirable visual phenomena will appear as a matter of course: for example, overlapping nodes, overlapping edges, nodes lying on top of edges – even in low stress drawings. An investigation of the (positive) effect of (invisible) stress on performance is also therefore an implicit investigation of the (negative) effect of these other consequent (visible) features.

Our stimuli were the same as those used for the previous Perception and Preference experiments; they were not specifically created for the purposes of this shortest path Performance experiment. The source and target nodes for the shortest paths were chosen randomly, ensuring an even distribution of path lengths across stress values. We could have created specific stimuli for the Performance experiment, and carefully chosen our shortest paths to avoid visual features that may have made the task easier or harder. However this would have resulted in an inauthentic, contrived set of drawings that may have been seen as deliberately formed so that they would support our intuition about the relationship between stress and task performance.

5.2.2 Demographics & Participant Strategies

A total of 36 participants completed the demographic questionnaire. Most participants reported feeling somewhat confident (30) in their responses, with only a few expressing very confident (1) or not very confident ratings (4). The majority found the study difficult (26) or very difficult (5), suggesting a reasonable level of challenge in the task. In terms of prior exposure, most were somewhat familiar (17) or not very familiar (14) with network diagrams, with only one participant indicating strong familiarity. There were 23 males and 13 females, and participants spanned a wide age range, with the largest groups falling between 26–35 (10) and 36–45 (9) years of age.

Of the 36 participants, 29 described the strategy they used to complete the task. As this was an optional, open-ended question, most of the responses were short or essentially restated the task ("I just tried to follow the lines", "I tried to look for the shortest paths", "Moving my mouse and counting").

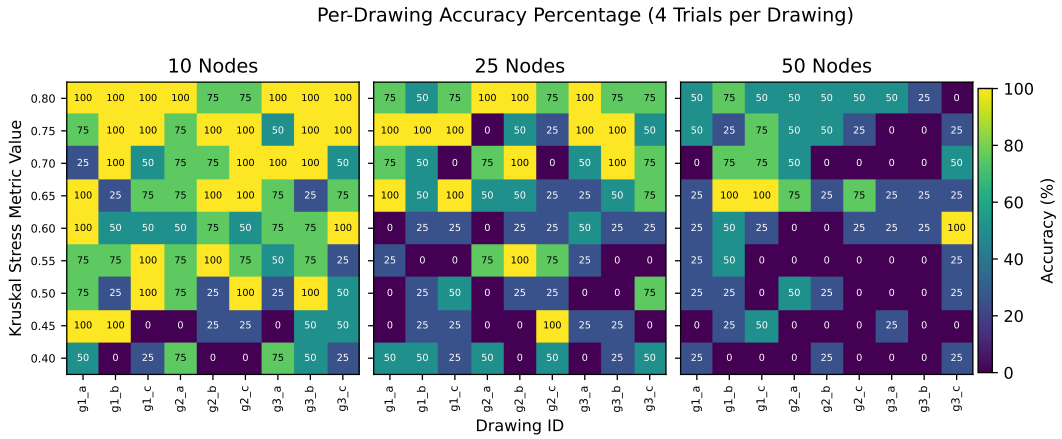


Figure 9 Per-drawing accuracy percentages (based on 4 trials per drawing) for each unique stimulus. Higher accuracy indicates better task performance. Each cell shows the mean accuracy (%) across participants for a specific drawing.

Several participants provided more detailed accounts of their strategies. Some began by scanning the whole diagram to "identify overall patterns" before following the "most intuitive connections." Others focused on counting steps, such as one who "looked for the shortest visual route between the two red nodes and counted the steps," or another who "tried to count the number of nodes in the shortest route" but sometimes found the diagrams too complex to work out. Some participants adapted their approach when a path became unclear: one reported they "tried from the other red dot" if their initial attempt failed, while another would abandon a path that took "over 5 hops" and try a different route. One participant highlighted the range of difficulties between trials: "Some tasks were easy to visualise. Some were impossible due to the complexity of lines making it difficult to see which node they terminated on." These responses indicate that while most participants relied on intuitive tracing or counting, several applied more reflective or adaptive techniques when faced with visual complexity.

6 Conclusions, Limitations and Future Work

Defining a normalised stress metric [24] that permits quantitative comparison between graph drawings of different size, structure and scale has allowed us to investigate human responses to stress in three ways: the perception of stress, preference for stress, and the effect of stress on task performance.

We conclude that people (even untrained novices) can perceive the extent of stress in a graph drawing, often using visible features (for example, "node distribution") as a proxy for the invisible "stress", while typically being unable to accurately define what stress means. People generally prefer low stress drawings, but making a preference choice is not the same as identifying the drawing with least stress. Lower stress drawings support higher accuracy in shortest path tasks.

Our conclusions are constrained by the natural limitations of the experimental method. We used three graph sizes for comparative and generalisability purposes, but the maximum graph size is only 50. Future experiments with larger graphs may have different findings. Our intuition is that as the size of the experimental graphs increase, the effect of stress (in all of perception, preference and performance) will decrease.

The graph drawings used as stimuli were generated using an optimisation function based only on the stress metric. Other metrics could conceivably be used, resulting in drawings that avoid negative features like node or edge overlaps. Since our focus was on stress, we did not want to create artificial experimental stimuli: changing stress naturally alters other features and so any (initial) investigation of stress should simply accept the existence of these features. Nevertheless, it would be interesting for future work to explore the effect of these other visual features on perception, preference, and performance.

The Stress Performance experiment is based on only a shortest-path task, and future work should investigate whether the same results hold true for other localised tasks (e.g., number of common neighbours of two nodes) as well as global tasks (e.g., extent of clustering). Our intuition is that, like with the shortest path task, lower stress will improve performance, as long as the graphs are not much bigger than 100 nodes – at which point we expect it will become difficult for change in stress to have any effect on human performance.

Our work represents the first comprehensive empirical investigation into the effect of stress in graph drawing from a human perspective. We have investigated whether people can actually perceive stress, what their preferences are with respect to stress, and whether the amount of stress affects task performance. For many years graph drawing researchers have made assumptions about the human response to stress: this paper validates these assumptions.

References

1 Reyan Ahmed, Felice De Luca, Sabin Devkota, Stephen G. Kobourov, and Mingwei Li. Multicriteria scalable graph drawing via stochastic gradient descent, $(SGD)^2$. *IEEE Transactions on Visualization and Computer Graphics*, 28(6):2388–2399, 2022. doi:10.1109/TVCG.2022.3155564.

2 Alessio Arleo, Silvia Miksch, and Daniel Archambault. Event-based dynamic graph drawing without the agonizing pain. *Computer Graphics Forum*, 41(6):226–244, 2022. doi:10.1111/cgf.14615.

3 Ulrik Brandes and Christian Pich. An experimental study on distance-based graph drawing. In *Graph Drawing*, volume 5417 of *Lecture Notes in Computer Science*, pages 218–229. Springer, 2008. doi:10.1007/978-3-642-00219-9_21.

4 Lisha Chen and Andreas Buja. Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis. *Journal of the American Statistical Association*, 104(485):209–219, 2009. doi:10.5555/2567709.2502616.

5 Markus Chimani, Patrick Eades, Peter Eades, Seok-Hee Hong, Weidong Huang, Karsten Klein, Michael Marner, Ross T. Smith, and Bruce H. Thomas. People prefer less stress and fewer crossings. In *Proceedings of International Symposium on Graph Drawing (GD)*, pages 523–524. Springer, 2014.

6 Sabin Devkota, Abu Reyan Ahmed, Felice De Luca, Katherine E. Isaacs, and Stephen G. Kobourov. Stress-Plus-X (SPX) graph layout. In *Graph Drawing and Network Visualization*, volume 11904 of *Lecture Notes in Computer Science*, pages 291–304. Springer, 2019. doi:10.1007/978-3-030-35802-0_23.

7 Sara Di Bartolomeo, Tarik Crnovrsanin, David Saffo, Eduardo Puerta, Connor Wilson, and Cody Dunne. Evaluating graph layout algorithms: A systematic review of methods and best practices. In *Computer Graphics Forum*, page e15073. Wiley Online Library, 2024.

8 Yiren Ding, Jack Wilburn, Hilson Shrestha, Akim Ndlovu, Kiran Gadhave, Carolina Nobre, Alexander Lex, and Lane Harrison. revisit: Supporting scalable evaluation of interactive visualizations. In *2023 IEEE Visualization and Visual Analytics (VIS)*, pages 31–35. IEEE, 2023. doi:10.1109/VIS54172.2023.00015.

9 Emden R. Gansner, Yifan Hu, and Stephen North. A maxent-stress model for graph layout. *IEEE Transactions on Visualization and Computer Graphics*, 19(6):927–940, June 2013. doi:10.1109/tvcg.2012.299.

10 Emden R. Gansner, Yehuda Koren, and Stephen C. North. Graph drawing by stress majorization. In *Graph Drawing and Network Visualization*, volume 3383 of *Lecture Notes in Computer Science*, pages 239–250. Springer, 2004. doi:10.1007/978-3-540-31843-9_25.

11 Seok-Hee Hong, Peter Eades, Marnijati Torkel, Ziyang Wang, David Chae, Sungpack Hong, Daniel Langerenken, and Hassan Chafi. Multi-level graph drawing using infomap clustering. In *Graph Drawing and Network Visualization*, volume 11904 of *Lecture Notes in Computer Science*, pages 139–146. Springer, 2019. doi:10.1007/978-3-030-35802-0_11.

12 Tomihisa Kamada and Satoru Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989. doi:10.1016/0020-0190(89)90102-6.

13 Johannes F. Kruiger, Paulo E. Rauber, Ricardo M. Martins, Andreas Kerren, Stephen Kobourov, and Alexandru C. Telea. Graph layouts by t-SNE. *Computer Graphics Forum*, 36(3):283–294, 2017. doi:10.1111/cgf.13187.

14 Joseph B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

15 Michael R. Marner, Ross T. Smith, Bruce H. Thomas, Karsten Klein, Peter Eades, and Seok-Hee Hong. GION: interactively untangling large graphs on wall-sized displays. In *Graph Drawing and Network Visualization*, volume 8871 of *Lecture Notes in Computer Science*, pages 113–124. Springer, 2014. doi:10.1007/978-3-662-45803-7_10.

16 Jacob Miller, Vahan Huroyan, and Stephen Kobourov. Balancing between the local and global structures (LGS) in graph embedding. In *International Symposium on Graph Drawing and Network Visualization*, pages 263–279. Springer, 2023. doi:10.1007/978-3-031-49272-3_18.

17 Gavin J. Mooney, Helen C. Purchase, Michael Wybrow, and Stephen G. Kobourov. The multi-dimensional landscape of graph drawing metrics. In *17th IEEE Pacific Visualization Conference, PacificVis 2024, Tokyo, Japan, April 23-26, 2024*, pages 122–131. IEEE, 2024. doi:10.1109/PACIFICVIS60374.2024.00022.

18 Gavin J. Mooney, Helen C. Purchase, Michael Wybrow, Stephen G. Kobourov, and Jacob Miller. The perception of stress in graph drawings. In *32nd International Symposium on Graph Drawing and Network Visualization (GD 2024)*, pages 21–1. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.GD.2024.21.

19 Networkx documentation. https://networkx.org/documentation/stable/reference/generated/networkx.drawing.layout.kamada_kawai_layout.html.

20 Mark Ortmann, Mirza Klimenta, and Ulrik Brandes. A sparse stress model. *Journal of Graph Algorithms and Applications*, 21(5):791–821, 2017. doi:10.7155/jgaa.00440.

21 Helen C. Purchase. Metrics for graph drawing aesthetics. *Journal of Visual Languages & Computing*, 13(5):501–516, 2002. doi:10.1006/jvlc.2002.0232.

22 John W Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, 100(5):401–409, 1969. doi:10.1109/T-C.1969.222678.

23 Paolo Simonetto, Daniel Archambault, and Stephen G. Kobourov. Drawing dynamic graphs without timeslices. In *Graph Drawing and Network Visualization*, volume 10692 of *Lecture Notes in Computer Science*, pages 394–409. Springer, 2017. doi:10.1007/978-3-319-73915-1_31.

24 Kiran Smelser, Jacob Miller, and Stephen Kobourov. "Normalized stress" is not normalized: How to interpret stress correctly. In *2024 IEEE Evaluation and Beyond-Methodological Approaches for Visualization (BELIV)*, pages 41–50. IEEE, 2024. doi:10.1109/BELIV64461.2024.00010.

25 Simon van Wageningen, Tamara Mchedlidze, and Alexandru Telea. An experimental evaluation of viewpoint-based 3D graph drawing. In *Computer Graphics Forum*, page e15077. Wiley Online Library, 2024. doi:10.1111/cgf.15077.

26 Xiaoqi Wang, Kevin Yen, Yifan Hu, and Han-Wei Shen. SmartGD: A GAN-Based graph drawing framework for diverse aesthetic goals. *IEEE Transactions on Visualization and Computer Graphics*, 2023. doi:10.1109/TVCG.2023.3306356.

27 Eric Welch and Stephen Kobourov. Measuring symmetry in drawings of graphs. *Computer Graphics Forum*, 36(3):341–351, June 2017. doi:10.1111/cgf.13192.

28 Vahan Yoghourdjian, Yalong Yang, Tim Dwyer, Lee Lawrence, Michael Wybrow, and Kim Marriott. Scalability of network visualisation from a cognitive load perspective. *IEEE transactions on visualization and computer graphics*, 27(2):1677–1687, 2020. doi:10.1109/TVCG.2020.3030459.

29 Minfeng Zhu, Wei Chen, Yuanzhe Hu, Yuxuan Hou, Liangjun Liu, and Kaiyuan Zhang. DRGraph: An efficient graph layout algorithm for large-scale graphs by dimensionality reduction. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1666–1676, 2021. doi:10.1109/TVCG.2020.3030447.

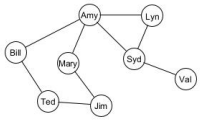
A Training Materials

Participants in the Perception experiment were shown training materials (1), (2) and (3). Participants in the Preference experiment were only shown training material (1). Participants in the Shortest Path experiment were shown training materials (1) and (2).

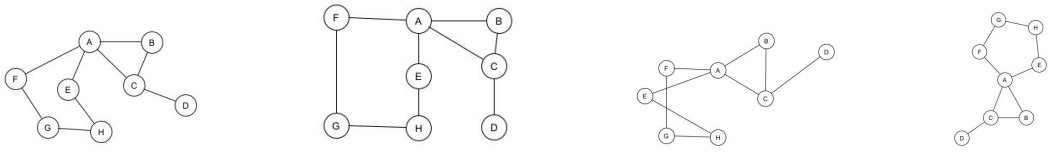
A.1 Training Material (1)

Definitions: networks and network drawings

A network is made up of objects and connections. For example, this social network depicts people (represented as circles) and friendships (represented as lines between the circles). Amy has four friends; Ted has two.

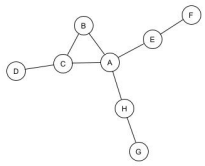


The same network can be drawn in many different ways by changing the position of the objects. For example, here are four drawings of the same network.



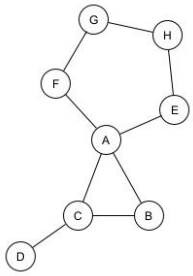
A.2 Training Material (2)

A "path" is a series of steps between objects. For example, in the network below, the length of the path between G and F is 4; the length of the path between A and D is 2 or 3 (depending on whether you go through B or not).



The "shortest path" is, as its name suggests, the shortest path when there is more than one way to get from one object to another.

For example, in the network below, the shortest path between F and B is 2 (going through A, but not C or G/H/E); the shortest path between D and H is 4 (going through C/A, but not through B or F/G).

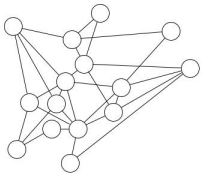


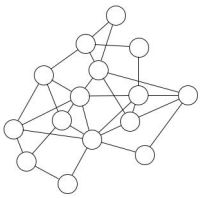
A.3 Training Material (3)

Definitions: visual properties of network drawings

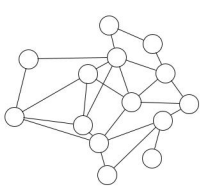
Given that there is more than one way to draw a network, we can distinguish between them by their 'visual properties'.

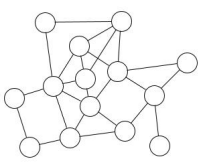
For example, the drawing on the left has 'tighter angles' than the one on the right. These are both drawings of the same network.





And the drawing on the right has "more symmetry" than the one on the left. These are both drawings of the same network.

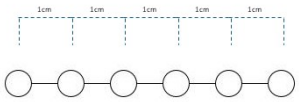




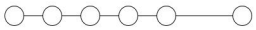
In this experiment, we are interested in the visual property of "stress".

A drawing has **low stress** if the distance between pairs of objects is proportional to the length of the shortest path between them.

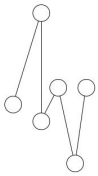
In its simplest form, the following network drawing has very low stress: the distance between each pair of objects is directly proportional to the (shortest) path between them.



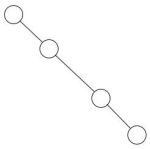
We just need to move one of the objects to **increase the stress** – the distance between the two objects at each end (6cm) is now longer proportional to the length of the path between them (5).

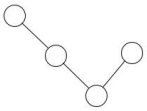


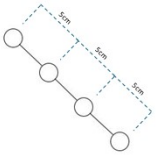
The same simple network can be drawn with even **higher stress**, where there is barely any relationship between the distance between the objects and the length of the paths between them.



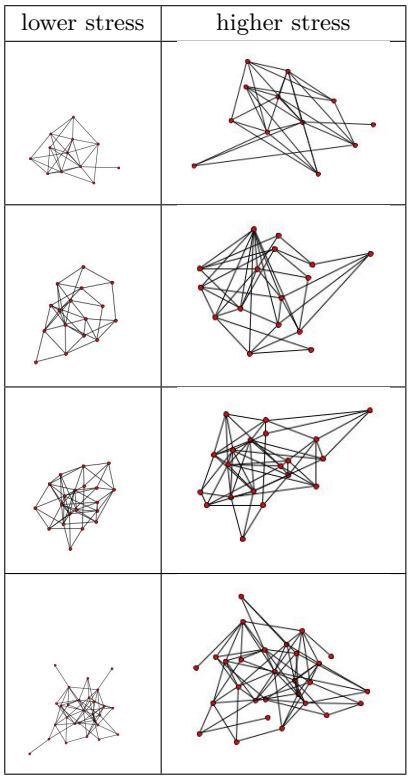
Similarly, here is another network with very low stress, with two versions of higher stress.







Of course, this more-or-less-stress judgement becomes more difficult with larger networks. Here are some more examples showing the same networks drawn with different amounts of stress. In all cases, the network on the left has less stress than the network on the right. Thus, the network on the left maintains the distance/path relationship between pairs of objects better than the one on the right.



Of course, we cannot assess the differences in stress by doing all the distance and path length calculations in our head! But we can get a 'feeling' as to when one drawing has less stress than another.

Before you start the experiment, we will ask you to make your own judgements, and let you know whether you are correct or not.

B Stimuli Metric Distributions and Correlations

We plot the distributions of 10 aesthetic metrics (defined by Mooney et al. [17]) for our 405 stimuli drawings (Figure 14). These distributions show that smaller graphs generally have a wider range of possible metric values. This can be explained by the fact that many of the metrics, such as Edge Crossings and Gabriel Ratio, compare a measurable value (e.g., the number of edge crossings) to an estimated maximum value (e.g., the total number of possible edge crossings). The estimated maximum typically grows much faster than the measured values as graph size increases.

C Detailed Analysis of the Difficulty of the Task for Each Drawing

The following heatmaps (Figures 15–23) show particular properties of each drawing with respect to the difficulty of the shortest path task. Higher row totals indicate greater difficulty for each KSM.

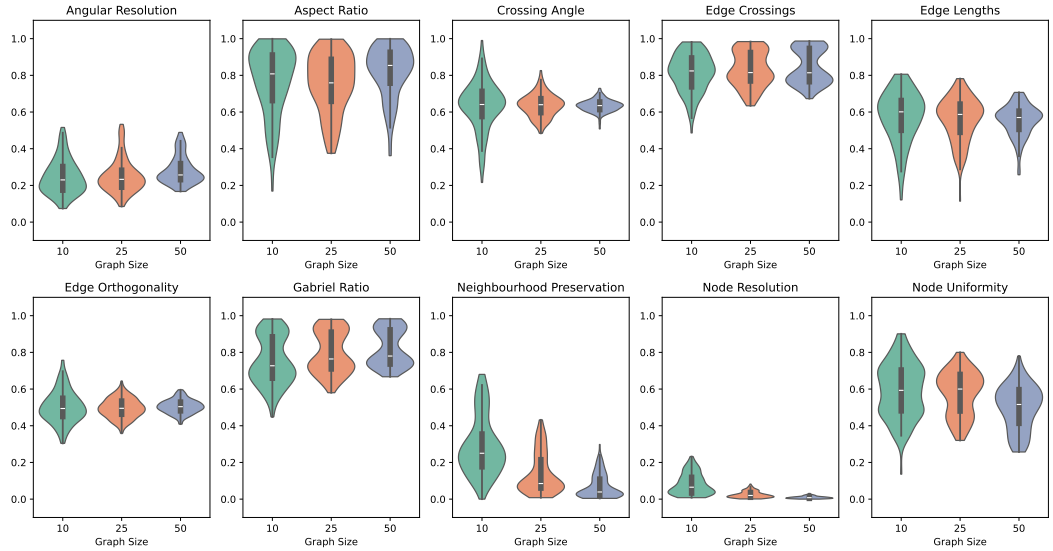


Figure 14 Distribution of metric values for the experiment stimuli, grouped by graph size.

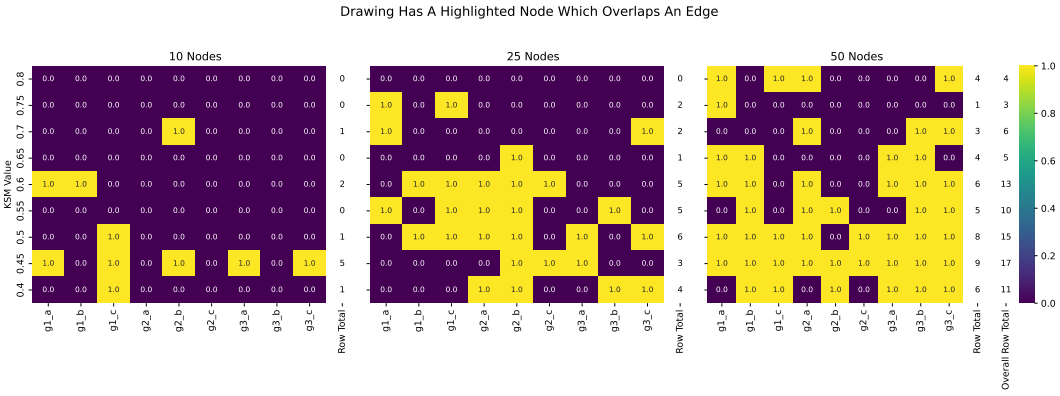


Figure 15 Drawings where a highlighted node overlaps another edge.

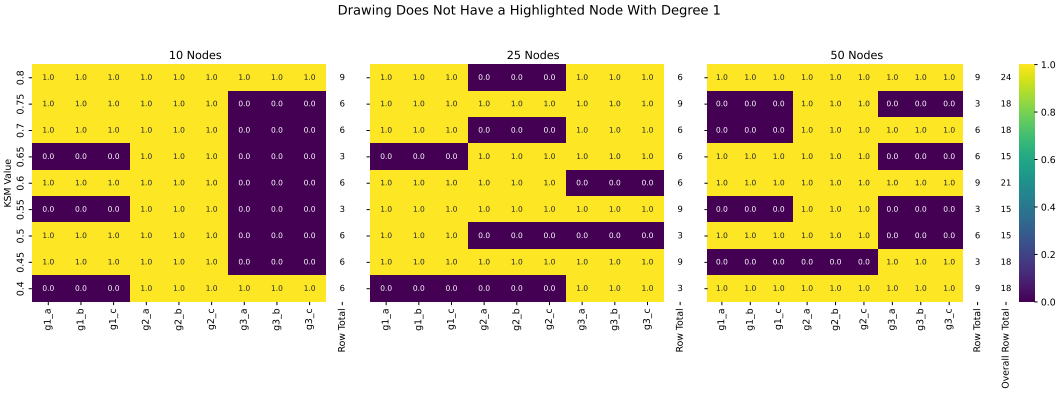


Figure 16 Drawings where no highlighted node has degree 1.

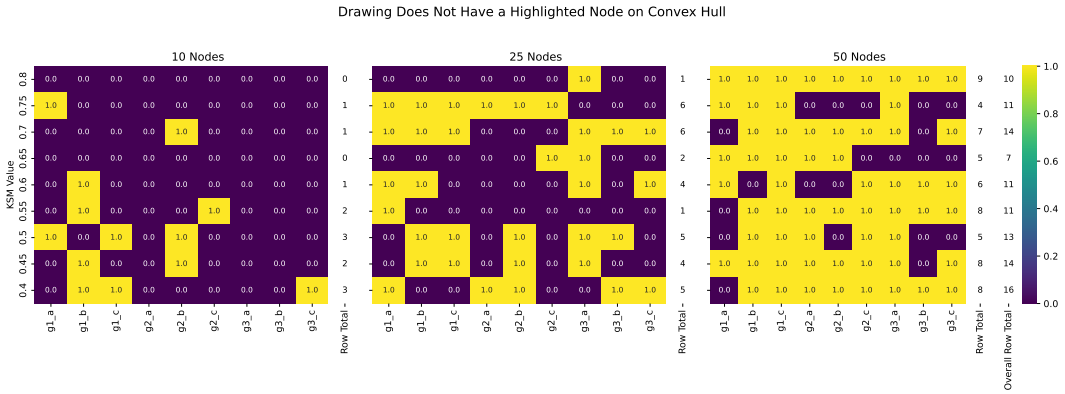


Figure 17 Drawings where no highlighted node is on the convex hull.

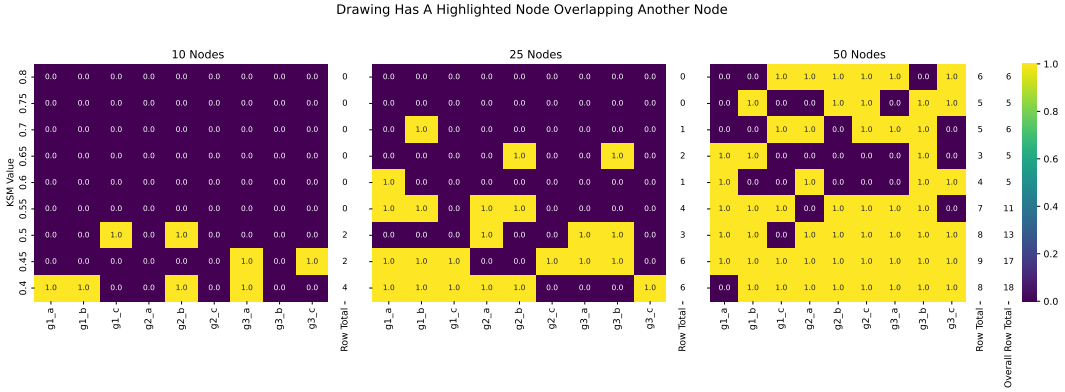


Figure 18 Drawings where a highlighted node overlaps another node.

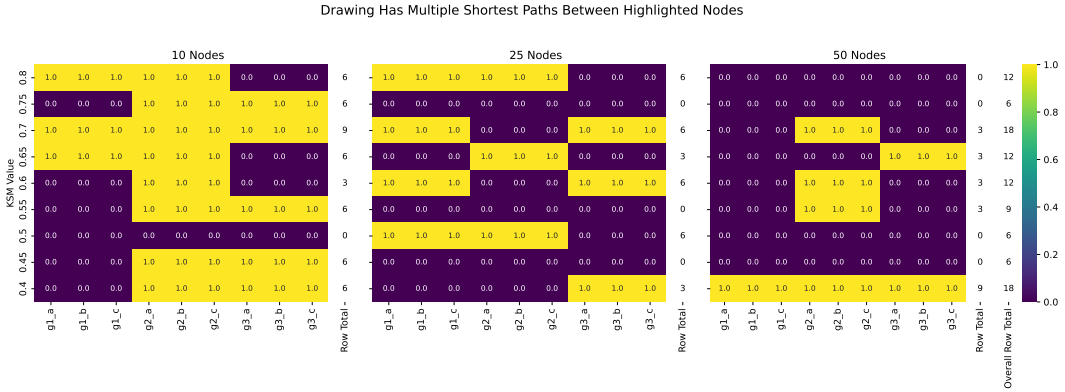


Figure 19 Drawings where there are multiple shortest paths between the highlighted nodes.

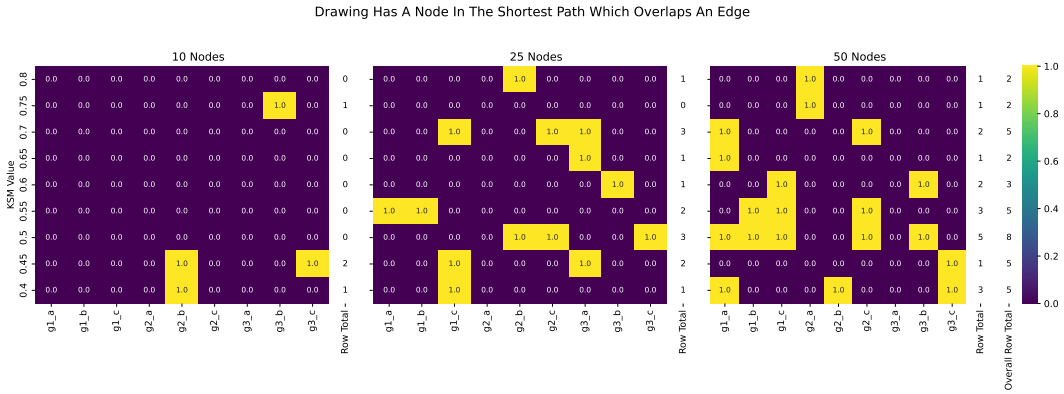


Figure 20 Drawings where a node on the shortest path overlaps an edge.

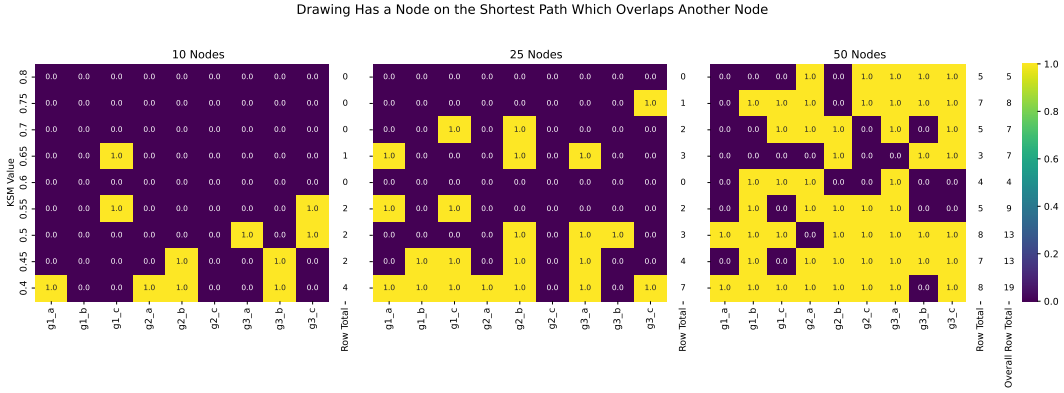


Figure 21 Drawings where a node on the shortest path overlaps another node.

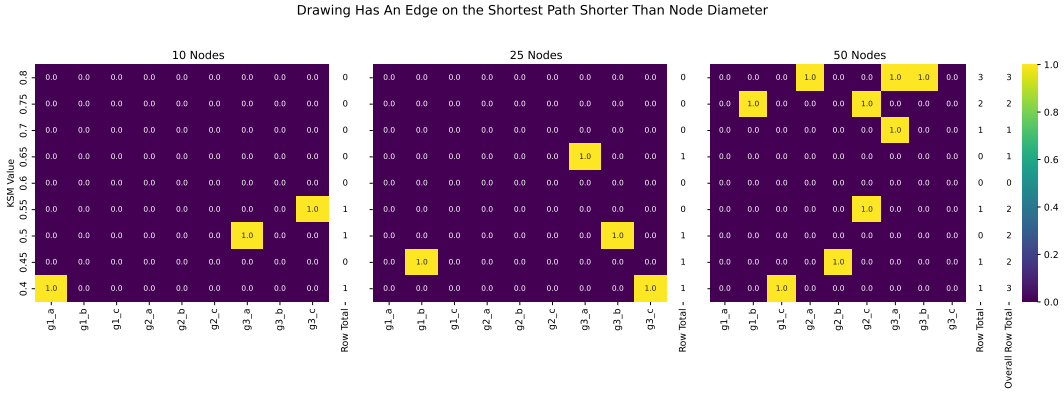


Figure 22 Drawings where an edge on the shorest path is shorter that the diameter of the nodes.

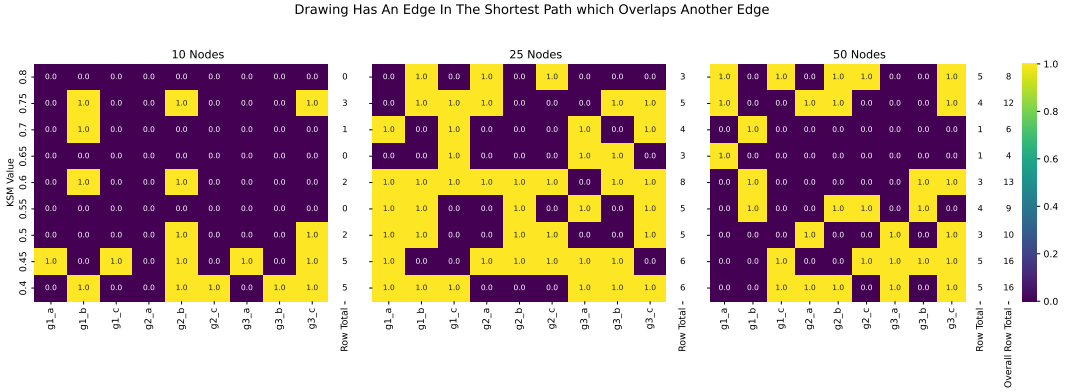


Figure 23 Drawings where an edge in the shorest path overlaps another edge.

Optimizing Wiggle in Storylines

Alexander Dobler
TU Wien, Austria

Tim Hegemann
Universität Würzburg, Germany

Martin Nöllenburg
TU Wien, Austria

Alexander Wolff
Universität Würzburg, Germany

Abstract

A *storyline visualization* shows interactions between characters over time. Each character is represented by an x-monotone curve. Time is mapped to the x-axis, and groups of characters that interact at a particular point t in time must be ordered consecutively in the y-dimension at $x = t$. The predominant objective in storyline optimization so far has been the minimization of crossings between (blocks of) characters. Building on this work, we investigate another important, but less studied quality criterion, namely the minimization of *wiggle*, i.e., the amount of vertical movement of the characters over time.

Given a storyline instance together with an ordering of the characters at any point in time, we show that *wiggle count minimization* is NP-complete. In contrast, we provide algorithms based on mathematical programming to solve *linear wiggle height minimization* and *quadratic wiggle height minimization* efficiently. Finally, we introduce a new method for routing character curves that focuses on keeping distances between neighboring curves constant as long as they run in parallel.

We have implemented our algorithms, and we conduct a case study that explores the differences between the three optimization objectives. We use existing benchmark data, but we also present a new use case for storylines, namely the visualization of rolling stock schedules in railway operation.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Human-centered computing → Graph drawings

Keywords and phrases Storyline visualization, wiggle minimization, NP-complete, linear programming, quadratic programming, experimental analysis

Digital Object Identifier 10.4230/LIPIcs.GD.2025.39

Related Version *Full Version*: `https://arxiv.org/abs/2508.19802` [7]

Supplementary Material *Software (Source Code)*: `https://github.com/hegetim/narrativiz` [14] archived at `swh:1:dir:a7efa9c0ca674b6cf25b1c4b2c1f4793cc814723`

Funding *Alexander Dobler*: Vienna Science and Technology Fund (WWTF) grant [10.47379/ICT19035]. *Tim Hegemann*: Federal Ministry of Research, Technology and Space (BMFTR) grant [01IS22012C]. *Martin Nöllenburg*: Vienna Science and Technology Fund (WWTF) grant [10.47379/ICT19035].

Acknowledgements We thank Marie Schmidt for discovering the similarities between Storylines and Rolling Stock Schedules. We thank Rowan Hoogervorst and Boris Grimm for providing the Rolling Stock Schedules dataset and for their helpful feedback.

1 Introduction

A *storyline* can be seen as a temporal hypergraph; the vertices represent *characters* and the hyperedges, which correspond to given points in time, represent *meetings* (also called *interactions*) among the characters. A *storyline visualization* draws each character as an

© Alexander Dobler, Tim Hegemann, Martin Nöllenburg, and Alexander Wolff;
licensed under Creative Commons License CC-BY 4.0

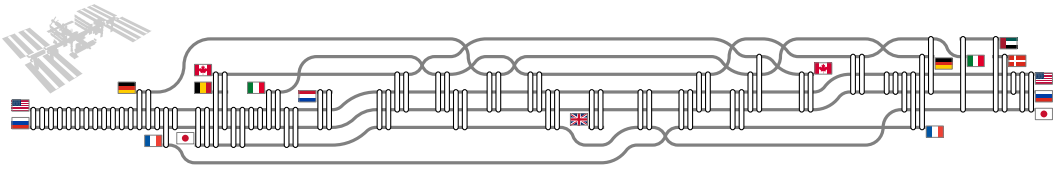


Figure 1 A storyline that visualizes the nationalities of the crew members of expeditions to the International Space Station. Linear wiggle height is minimized using our LP formulation.

x-monotone curve and each meeting as a vertical line segment at the x-coordinate that corresponds to the point in time when the meeting happens; see Figures 1 and 2a. Storyline visualizations have been made for books or movies [8, 13, 20, 22, 26] (where the meetings are the scenes of the story), but they have also been used to illustrate scientific collaboration [15] (where the meetings are joint publications), for genealogical data [17] (where the meetings are marriages and have size 2), or for tweets following certain topics [20].

In order to measure the quality of storyline visualizations, various metrics have been suggested. Most works have focused on reducing the number of crossings of the character curves (simple pairwise crossings [13, 19] or so-called block crossings [27, 28]). Others have also tried to reduce the number of wiggles [11, 20, 26] (that is, the number of turns) and/or the amount of vertical white-space [20, 26].

In this paper, we follow the storyline layout pipeline that has been suggested by Liu, Wu, Wei, Liu, and Liu [20, Fig. 2]. It breaks down the overall problem into four steps; 1. hierarchy generation, 2. ordering (crossing minimization), 3. alignment (wiggle minimization), and 4. compaction (white-space reduction). For step 1, Liu et al. assume that the characters form a given hierarchy that must be respected in step 2. We assume that steps 1 and 2 have been settled, optimally or heuristically. Note that crossing minimization is NP-hard [19, 27].

This paper focuses on wiggle minimization, which appears in step 3 and, in a different form, in step 4. We differentiate between three variants; in each of them, we are given, for every point in time, the vertical ordering of the characters, which for us is fixed. In *wiggle count minimization* (WCMin) the task is to find, for each point in time, y-coordinates for the character curves such that the total number of inflection points (points where the curvature of the character curves changes sign) is minimized. In *linear wiggle height minimization* (LWHMin), the total change in y-coordinate is minimized (see the example in Figure 1). Finally, in *quadratic wiggle height minimization* (QWHMin), the total sum of the squared wiggle heights is minimized.

Related work. In terms of wiggle minimization in storylines, the existing literature mostly proposes heuristic methods, does not formally define a wiggle metric, or considers models that differ from our setting. Ogawa and Ma [23] propose a greedy algorithm for storyline visualization that attempts to simultaneously minimize crossings and wiggle, however, wiggle is not formally defined. Tanahashi and Ma [26] present a genetic algorithm for drawing storylines. They make use of *slots*, which are horizontal strips of the visualization. A genome assigns each meeting to a slot. To evaluate a genome a pipeline approach is used, which rearranges lines inside slots, and then computes a fitness function measuring crossings and wiggle. Liu et al. [20] also present a pipeline approach for drawing storylines, where characters further have geographic locations. Crossings are minimized first by applying a variant of the barycenter heuristic [24] sequentially, then wiggle count is minimized heuristically by reducing the problem to the weighted longest common subsequence problem of two neighboring time steps. Lastly, quadratic wiggle height and whitespace is minimized using a quadratic program. The authors of [25] generalize the earlier approach of Tanahashi and Ma [26] to streaming

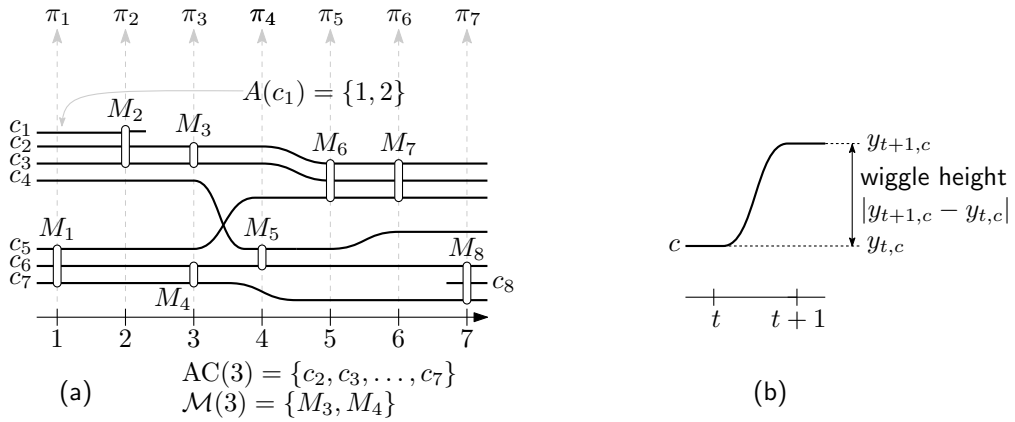


Figure 2 (a) Notation of storylines. (b) Wiggle height of character c between two time steps.

data, where time steps appear one by one. Arendt and Pirrung [1] consider a model where characters can split and merge. They first minimize crossings, and then reduce wiggle count minimization to the independent set problem, which they solve heuristically. Fröschl and Nöllenburg [10, 11] proposed a model for storyline visualization where characters are assigned to discrete cells in a matrix for each time step. They used ILP and SAT solvers to simultaneously minimize a combination of crossings, wiggle count, and linear wiggle height.

Wiggle minimization also plays a role in *streamgraphs* and *stacked area charts* [2, 5, 21], which are visualizations of multiple time series vertically stacked on top of each other without gaps. In these charts, wiggle is minimized combinatorially by computing the best stacking order of the time series.

Our contribution. is as follows.

- We present a linear program (LP) to solve LWHMin efficiently; see Section 3.1.
- We give a quadratic program (QP) to solve QWHMin efficiently; see Section 3.2.
- We prove that WCMin is NP-hard, but can be formulated as an integer linear program (ILP) and admits an efficient solution for two consecutive time steps; see Section 4.
- We present a new method for routing character curves that focuses on keeping distances between neighboring curves constant as long as they run in parallel; see Section 5.
- We report the results of a case study with 17 benchmark instances in which we compare optimal solutions for the three objectives qualitatively and quantitatively; see Section 6.1.
- We present the visualization of rolling stock schedules as a new use case for storylines; see Section 6.2. This can help railway experts to improve or compare such schedules.

Due to space constraints, statements marked with ($\star$) are proved in the full version of this paper [7].

2 Preliminaries

We use $[n]$ as shorthand for $\{1, 2, \ldots, n\}$. A *storyline instance* is a 4-tuple $(\mathcal{C}, T, \mathcal{M}, A)$, where $\mathcal{C} = \{c_1, \ldots, c_n\}$ is a set of *characters*, $T = [\ell]$ is a set of totally ordered *time steps* (or *layers*), and $\mathcal{M} = \{M_1, \ldots, M_m\}$ is a set of *meetings*; see Figure 2a. Each meeting $M \in \mathcal{M}$ has a corresponding time step $\mathrm{time}(M) \in [\ell]$ and consists of a set of characters $\mathrm{ch}(M) \subseteq \mathcal{C}$. Each character $c \in \mathcal{C}$ is *active* for a sequence $A(c) = \{i, i+1, \ldots, j\}$ of consecutive time steps. For each $t \in [\ell]$, we define the active character set $\mathrm{AC}(t) = \{c \in \mathcal{C} \mid t \in A(c)\}$ and the meeting set $\mathcal{M}(t) = \{M \in \mathcal{M} \mid \mathrm{time}(M) = t\}$.

An *ordered storyline instance* is furthermore given, for each $t \in [\ell]$, a permutation π_t of the characters $\mathrm{AC}(t)$. Here, for each meeting M with $\mathrm{time}(M) = t$, the characters $\mathrm{ch}(M)$ appear consecutively in π_t. We write $c \prec_t c'$ if c comes before c' in π_t. A *coordination* of a storyline instance defines for each t and each $c \in \mathrm{AC}(t)$ a y-coordinate $y_{t,c} \in \mathbb{R}$. The coordination is *valid* w.r.t. an ordered storyline instance if for each $t \in [\ell]$ and each pair $c, c' \in \mathrm{AC}(t)$ with $c \prec_t c'$, $y_{t,c} < y_{t,c'}$ holds. Given $\Delta, \overline{\Delta} \in \mathbb{R}_0^+$, a coordination is $(\Delta, \overline{\Delta})$-*nice* (or *nice* if Δ and $\overline{\Delta}$ are clear from the context) if it is valid and if, for every $t \in [\ell]$ and for every pair of characters c, c' that are consecutive in π_t, it holds that

- $|y_{t,c'} - y_{t,c}| = \Delta$ if c and c' are in the same meeting at time step t, and
- $|y_{t,c'} - y_{t,c}| \geq \overline{\Delta}$ otherwise.

A coordination is *integral* if its image contains only integers.

Given a coordination y of an ordered storyline instance, its *total (linear) wiggle height* (see Figure 2b) is defined as

$$\mathrm{LWH}(y) = \sum_{t=1}^{\ell-1} \sum_{c \in \mathrm{AC}(t) \cap \mathrm{AC}(t+1)} |y_{t,c} - y_{t+1,c}|. \tag{1}$$

Its *total quadratic wiggle height* is defined as

$$\mathrm{QWH}(y) = \sum_{t=1}^{\ell-1} \sum_{c \in \mathrm{AC}(t) \cap \mathrm{AC}(t+1)} (y_{t,c} - y_{t+1,c})^2. \tag{2}$$

Finally, its *total wiggle count* is defined as

$$\mathrm{WC}(y) = \left|\{(t,c) \mid 1 \leq t \leq \ell - 1, c \in \mathrm{AC}(t) \cap \mathrm{AC}(t+1), y_{t,c} \neq y_{t+1,c}\}\right|. \tag{3}$$

Note that this matches our intuitive definition of wiggles as inflection points of the character curves.

▶ **Problem 1** (LWHMin)**.** Given an ordered storyline instance and $\Delta, \overline{\Delta} \in \mathbb{R}$, find a nice coordination minimizing the total wiggle height.

▶ **Problem 2** (QWHMin)**.** Given an ordered storyline instance and $\Delta, \overline{\Delta} \in \mathbb{R}$, find a nice coordination minimizing the total quadratic wiggle height.

▶ **Problem 3** (WCMin)**.** Given an ordered storyline instance and $\Delta, \overline{\Delta} \in \mathbb{R}$, find a nice coordination minimizing the total wiggle count.

3 Wiggle Height Minimization

In this section, we describe a linear program (LP) for LWHMin and a quadratic program (QP) for QWHMin. We also present observations about these programs.

3.1 Linear Program for LWHMin

Note that linear programs are polynomial-time solvable [16]. The linear program makes use of the following variables.

- for $t \in [\ell], c \in \mathrm{AC}(t)$, $y_{t,c}$ encodes the y-coordinate of character c at time step t.
- for $t \in [\ell - 1], c \in \mathrm{AC}(t) \cap \mathrm{AC}(t+1)$, $w_{t,c}$ encodes the wiggle height of character c between time steps t and $t+1$.

For our LP formulation we define, for each $t \in [\ell]$, the sets

$$N(t) = \{(c, c') \mid c \text{ and } c' \text{ are characters that are consecutive in } \pi_t \text{ and } c \prec_t c'\},$$
$$N_{\mathcal{M}}(t) = N(t) \cap \{(c, c') \mid c \text{ and } c' \text{ are in the same meeting at time step } t\}, \text{ and}$$
$$N_A(t) = N(t) \setminus N_{\mathcal{M}}(t)$$

The LP is given as follows.

$$\text{Minimize} \sum_{t \in [\ell], c \in \mathrm{AC}(t)} w_{t,c}$$

$$\begin{aligned}
\text{subject to} \quad y_{t,c'} - y_{t,c} &= \Delta && \text{for all } t \in [\ell], (c, c') \in N_{\mathcal{M}}(t) && \text{(Cons-}\Delta\text{)}\\
y_{t,c'} - y_{t,c} &\geq \overline{\Delta} && \text{for all } t \in [\ell], (c, c') \in N_A(t) && \text{(Cons-}\overline{\Delta}\text{)}\\
y_{t,c} - y_{t+1,c} &\leq w_{t,c} && \text{for all } t \in [\ell - 1], c \in \mathrm{AC}(t) \cap \mathrm{AC}(t+1) && \text{(W1)}\\
y_{t+1,c} - y_{t,c} &\leq w_{t,c} && \text{for all } t \in [\ell - 1], c \in \mathrm{AC}(t) \cap \mathrm{AC}(t+1) && \text{(W2)}\\
y_{t,c} &\geq 0 && \text{for all } t \in [\ell], c \in \mathrm{AC}(t) && \text{(G0)}
\end{aligned}$$

Constraints (Cons-$\overline{\Delta}$) and (Cons-Δ) ensure that the computed coordination is $(\Delta, \overline{\Delta})$-nice. Together with the objective, (W1) and (W2) ensure that the wiggle value is computed correctly as $w_{t,c} = |y_{t,c} - y_{t+1,c}|$. The objective adds up the wiggle values. Our LP has the following nice property.

▶ **Proposition 1.** *If $\Delta, \overline{\Delta} \in \mathbb{N}$, then all extreme points defined by the wiggle height minimization polytope are integer.*

Proof. Assume assignments to y- and w-variables corresponding to an extreme point. It is clear that $\min\{y_{t,c} \mid t \in [\ell], c \in \mathrm{AC}(t)\} = 0$, as otherwise there is some small $\epsilon > 0$ such that each y-variable can be simultaneously increased or decreased by ϵ, contradicting that we are at an extreme point. It is also clear that, for each $t \in [\ell-1]$ and $c \in \mathrm{AC}(t) \cap \mathrm{AC}(t+1)$, either (W1) or (W2) is binding. Now, let G be the graph defined by $V(G) = \{y_{t,c} \mid t \in [\ell], c \in \mathrm{AC}(t)\}$ and $E(G) = \{y_{t,c}y_{t',c'} \mid y_{t,c} - y_{t',c'} \in \mathbb{Z}\}$. If G is connected, we are done as we have already seen that the smallest y-variable is integer. Otherwise, consider some connected component C of G such that $V(C) \not\subseteq \mathbb{Z}$. Note that it is possible to add to all $y_{t,c} \in V(C)$ some small $\epsilon > 0$, resulting in a different point enclosed in the polytope. Equivalently, there exists $\epsilon' > 0$ that can be subtracted from all $y_{t,c} \in V(C)$, witnessing in fact that we are not at an extreme point. Thus, G is connected and the extreme point is integer. ◀

3.2 Quadratic Program for QWHMin

The quadratic program makes use of the same variables and constraints as the LP above. In fact, we just replace the objective function by the definition of quadratic wiggle height

$$\text{Minimize} \sum_{t \in [\ell], c \in \mathrm{AC}(t)} w_{t,c}^2.$$

It is easy to see that, in matrix form, the objective has diagonal entries that are either 0 or $w_{t,c}$ (for some $t \in [\ell - 1]$ and $c \in \mathrm{AC}(t)$). The entries of type $w_{t,c}$ are non-negative by definition, hence the matrix of the objective is positive semidefinite. QPs with such an objective and with linear constraints can be solved in polynomial time [4].

Note that we can replace each of the summands $w_{t,c}^2$ in the objective by $(y_{t,c} - y_{t+1,c})^2$, which is the same as $(y_{t+1,c} - y_{t,c})^2$ (for all $t \in [\ell - 1], c \in \mathrm{AC}(t) \cap \mathrm{AC}(t + 1)$). Then we can drop the w-variables and the constraints (W1) and (W2).

4 Wiggle Count Minimization

In this section we consider the number of wiggles in the visualization as optimization criterion. We show that for $\ell = 2$ time steps, WCMin is polynomial-time solvable, while it is NP-hard in the general case. We also present an integer linear program to solve WCMin.

4.1 Polynomial Cases

When ignoring the restriction that characters in meetings need to be equally spaced, finding a storyline visualization with the fewest number of wiggles is polynomial-time solvable. The problem is equivalent to finding the longest common subsequence between each pair of adjacent orderings π_t, π_{t+1}, $t \in [\ell - 1]$. Integer coordinates can be obtained by scaling all y-coordinates accordingly. A similar observation has been made in [20].

▶ **Observation 2.** *Given an ordered storyline instance, one can compute in polynomial time a valid integral coordination that minimizes the total wiggle count.*

Rather surprisingly, we obtain the following result. It shows that we can compute the "maximum wiggle-free" storyline – a set of characters that all can be realized without a single wiggle – in polynomial time. The proof is in the full version [7].

▶ **Theorem 3** ($\star$). *Let $(\mathcal{C}, T, \mathcal{M}, A, (\pi_t)_{t\in[\ell]})$ be an ordered storyline instance. Let $\mathcal{C}'$ be the set of characters in $\mathcal{C}$ that are active at all time steps, i.e., $A(c) = [\ell]$ for all $c \in \mathcal{C}'$. Then, given a pair $(\Delta, \overline{\Delta})$, a $(\Delta, \overline{\Delta})$-nice coordination with the largest wiggle-free subset of $\mathcal{C}'$ can be computed in $\mathcal{O}(|\mathcal{C}|^2 \cdot \ell)$ time.*

The next result follows directly, as for $\ell = 2$, the only characters that can have (at most one) wiggle, are active at both time steps.

▶ **Corollary 4.** WCMin *is solvable in time $\mathcal{O}(|\mathcal{C}|^2)$ for $\ell = 2$.*

4.2 NP-Hardness

In this section, we prove the following result with a gadget-based reduction. We first describe the source problem of the reduction, then the required gadgets, and finally the full reduction.

▶ **Theorem 5.** *The decision variant of* WCMin *is* NP*-complete.*

Our proof reduces from the problem Planar Monotone 3-Sat, which is defined as follows. Let ϕ be a Boolean 3-Sat formula in conjunctive normal form, let Υ be the set of variables of ϕ, and let Γ be the set of clauses of ϕ. The formula ϕ is *monotone* if every clause of ϕ

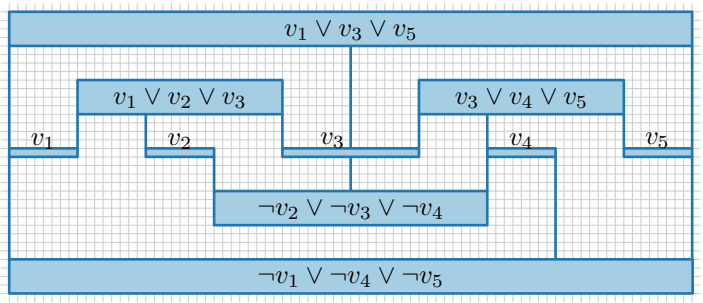


Figure 3 An instance of PM3-Sat.

contains either only positive or only negative literals. We define the bipartite variable-clause graph $G(\phi)$, consisting of vertex set $\Gamma \cup \Upsilon$ and an edge between a clause $\gamma \in \Gamma$ and a variable $v \in \Upsilon$ if and only if v occurs in c (positively or negatively). A *monotone planar rectilinear embedding* $\mathcal{E}$ of $G(\phi)$ is a planar embedding of $G(\phi)$ on an integer grid (see Figure 3) such that

- all vertices in $G(\phi)$ are represented by disjoint integer coordinate rectangles,
- the y-coordinates of all variable vertices are the same,
- all positive clause vertices are placed above the variable vertices,
- all negative clause vertices are placed below the variable vertices, and
- edges are straight vertical segments with integer x-coordinates connecting the borders of their corresponding rectangles.

▶ **Problem 4** (Planar Monotone 3-Sat (PM3-Sat))**.** Given a monotone 3-Sat formula ϕ and a monotone planar rectilinear embedding $\mathcal{E}$ of $G(\phi)$, decide whether ϕ is satisfiable.

PM3-Sat is NP-complete [6]. We can assume w.l.o.g. that the coordinates in the embedding $\mathcal{E}$ have the following special properties:

- The x-coordinates of edges and vertical borders of the vertex rectangles are multiples of 8.
- All edge lengths are multiples of 4.
- The clause vertices have height 4 and the variable vertices have height 1.

Notice that Figure 3 shows an instance satisfying these properties.

Now, given such an instance $I_{\text{SAT}} = (\phi, \mathcal{E})$ of PM3-Sat, we construct an instance $I_{\text{SL}} = (\mathcal{C}, T, \mathcal{M}, A, (\pi_t)_{t\in[\ell]}, \Delta, \overline{\Delta})$ of WCMin. We will give the precise number of wiggles W for the decision variant of WCMin later, and set $\Delta = \overline{\Delta} = 1$, thus, we can assume that coordinations of I_{SL} contain only integer coordinates (see Lemma 11).

To have a better correspondence between the two instances, we assume in the reduction that the storyline curves are not x-monotone, but are y-monotone from bottom to top. That is, we rotate the storyline drawing by 90 degrees counterclockwise. All figures for the reduction reflect this. We assume w.l.o.g. that the smallest y-coordinate of any vertex in $\mathcal{E}$ is 3. Let $y_{\max}$ be the largest y-coordinate. We set $T = \{1, \dots, y_{\max} + 2\}$. The main part of our reduction will be in $[3, y_{\max}] \subseteq T$, while the remaining four time steps will be used for an auxiliary bounding *frame*.

Our reduction consists of three components: variable gadgets, clause gadgets, and a rigid frame. For presentation purposes, we define the storyline instance graphically. That is, we define the instance in terms of one of its coordinations. The definitions of $\mathcal{C}, \mathcal{M}, A$, and $(\pi_t)_{t\in[\ell]}$ can then be inferred from this coordination. In the main part of the paper, we assume a black-box bounding frame, which we assume to have fixed coordinates in any coordination. Clearly, this frame must consist of meetings and characters, its realization is given in in the full version [7] due to space constraints. An instance of WCMin that results from the instance in Figure 3 can already be found in Figure 6 in the variant with a black-box bounding frame. We start with the description of the variable gadget.

Variable gadget. Assume all variables in $\mathcal{E}$ are embedded on the two y-coordinates y_{top} and y_{bot}. Now consider a variable $v \in \Upsilon$. We assume that the corresponding rectangle in $\mathcal{E}$ has width w_v, and its horizontal edges span from x_{left} to x_{right}. Then the corresponding variable gadget (see Figure 4) consists of a set C_v of w_v characters which are ordered as in Figure 4 and are active at time steps y_{bot} and y_{top}. This set of characters can be further partitioned into a set C_v^{pos} of *positive characters* (green in Figure 4) and a set C_v^{neg} of *negative characters* (red in Figure 4). Furthermore, using a frame as in the figure, the variable gadget is confined to a space of width $w_v + 2$, with coordinates from $x_{\text{left}} - 1$ to $x_{\text{right}} + 1$.

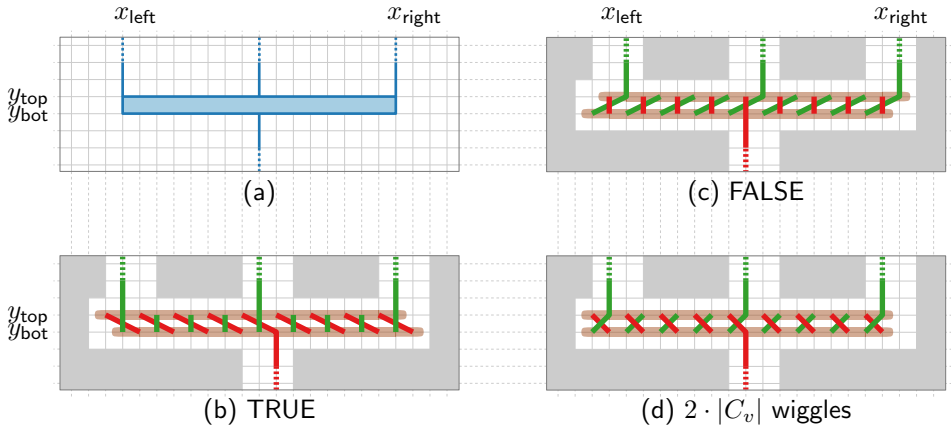


Figure 4 Illustration of a variable gadget. (a) The variable in the embedding $\mathcal{E}$. (b) The corresponding variable gadget in the TRUE state. (c) The variable gadget in the FALSE state. (d) The variable gadget in a state with too many wiggles. Character curves are green and red. meetings are brown horizontal blocks. The rigid frame is shown in gray. The dotted lines sketch the connections to the corresponding clause gadgets.

For each occurrence of a variable in a clause, a character is extended towards a clause gadget, as indicated with the dashed line ends in Figure 4. Positive characters are connected with positive clauses, negative characters with negative clauses. More precisely, for a vertical edge of $\mathcal{E}$ connecting v with a positive (negative) clause $\gamma \in \Gamma$ at x-coordinate $x_{\text{left}} + x_{v,\gamma}$ with $x_{v,\gamma} \in \mathbb{N}_0$, let $c_{v,\gamma} \in C_v^{\text{pos}}$ ($c_{v,\gamma} \in C_v^{\text{neg}}$) be the $(1+x_{v,\gamma}/2)$th (from left to right) positive (negative) character in C_v^{pos} (C_v^{neg}) (the values are always positive as x-coordinates of edges are multiples of eight). We call such a character *wire character*. This character is extended to time steps $y_{\text{top}}+1, y_{\text{top}}+2, \dots$ ($y_{\text{bot}}-1, y_{\text{bot}}-2, \dots$) until it reaches the corresponding clause gadget. This connection is along a corridor of width two defined by x-coordinates $x_{\text{left}} + x_{v,\gamma}$ and $x_{\text{left}} + x_{v,\gamma} + 1$ (see Figure 4). In a potential coordination, we say that $c_{v,\gamma}$ is *satisfying* if the x-coordinate of $c_{v,\gamma}$ at y_{top} (y_{bot}) is $x_{\text{left}} + x_{v,\gamma}$, and not satisfying otherwise. We say that $x_{\text{left}} + x_{v,\gamma}$ is the *satisfying x-coordinate* of $c_{v,\gamma}$.

Due to the fixed coordinates of frames, we observe the following.

▶ **Observation 6.** *The variable gadget has either $|C_v|$ (Figure 4b,c) or $2|C_v|$ (Figure 4d) wiggles between time steps y_{top} and y_{bot}.*

Furthermore, as desired, variable gadgets model a choice between TRUE and FALSE (see Figure 4b,c).

▶ **Observation 7.** *If the variable gadget has $|C_v|$ wiggles, then either all positive wire characters are satisfying and all negative wire characters are not satisfying, or vice versa.*

Clause gadget. At the heart of the reduction is the clause gadget, see Figure 5. We desribe the gadget for positive clauses only. The gadgets for negative clauses are the same but mirrored along the x-axis and can also be found in Figure 6. Thus, let γ be a positive clause with three variables $v_1, v_2, v_3 \in \Upsilon$, such that v_1 is the left, v_2 is the middle, and v_3 is the right variable. Assume that the clause rectangle in $\mathcal{E}$ is defined by the coordinates $[x_{\text{left}}, x_{\text{right}}] \times [y_{\text{bot}}, y_{\text{top}}]$. The clause gadget for γ is essentially placed into the rectangle

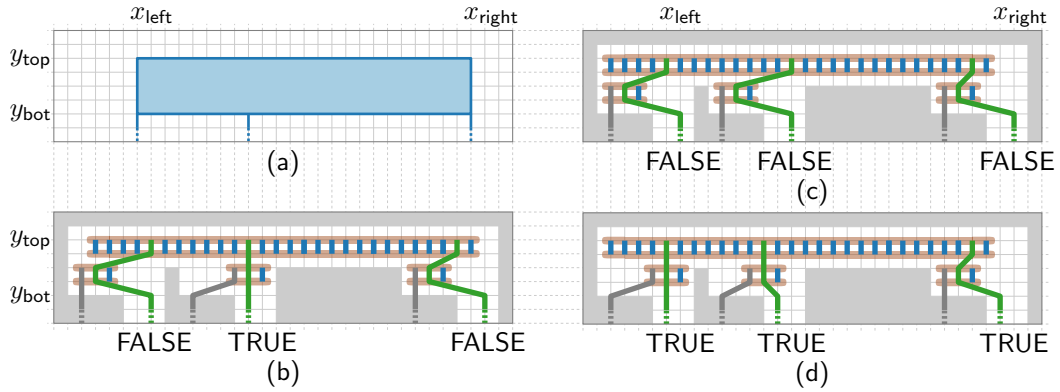


Figure 5 Illustration of a positive clause gadget. (a) The clause in the embedding $\mathcal{E}$. (b-d) The same clause gadget in three different states depending on whether its wire characters are satisfying (TRUE) or not (FALSE). A character belonging to the rigid frame is shown with the gray line.

$[x_{\text{left}} - 4, x_{\text{right}} + 1] \times [y_{\text{bot}}, y_{\text{top}}]$.[1] The rest of the clause gadget depends on the satisfying x-coordinates of the characters $c_{v_i,\gamma}$ with $i \in [3]$; see Figure 5. Thus, let x_1^s, x_2^s, x_3^s be the satisfying x-coordinates of $c_{v_1,\gamma}, c_{v_2,\gamma}, c_{v_3,\gamma}$, respectively. Each of the three wire characters is active until time step y_{top}, and it is part of some meetings with *clause gadget characters* (blue in Figure 5). For each wire character $c_{v_i,\gamma}$, we have the following construction.

- A *fixing character* which is a character from the frame fixed at $x_i^s - 4$ which is extended until $y_{\text{bot}} + 2$, see the gray line in Figure 5.
- The two *blocking meetings* of size three at time steps $y_{\text{bot}} + 1$ and $y_{\text{bot}} + 2$ that ensure that $c_{v_i,\gamma}$ must have at least one wiggle if its x-coordinate at y_{bot} is not satisfying.

Lastly, the clause gadget has two *choice meetings* of size $x_{\text{right}} - x_{\text{left}} + 4$ at y_{top} and $y_{\text{top}} - 1$. Each clause character $c_{v_i,\gamma}$ with $i \in [3]$ appears inside these meetings at position $x_i^s - x_{\text{left}} + i$. The remaining positions are filled by clause characters which have the same position in both meetings. Essentially, the addition of i, which slightly shifts the relation of satisfying x-coordinate and position in the meeting, ensures that we cannot prevent multiple wiggles by having multiple satisfying wire characters. The following two lemmas show that the clause gadget can be realized with the fewest wiggles if at least one of its characters is satisfying, and otherwise with one wiggle more. We assume that wire characters do not have wiggles between their variable gadget and y_{bot}, as they are either unnecessary or can be moved into the clause gadget.

▶ **Lemma 8.** *Assume that there exists a nice coordination in which none of $c_{v_1,\gamma}$, $c_{v_2,\gamma}$, and $c_{v_3,\gamma}$ is satisfying. Then a nice coordination of the clause gadget has at least six wiggles, and such a coordination with exactly six wiggles always exists.*

Proof. Figure 5 depicts such a nice coordination. We show that the number of wiggles of each wire character $c_{v_i,\gamma}$ with $i \in [3]$ and its corresponding fixing character is at least two between y_{bot} and y_{top}. If the number of wiggles is at least two between y_{bot} and $y_{\text{bot}} + 1$, we are done.

Otherwise, $c_{v_i,\gamma}$ has x-coordinate $x_i^s - 3$ at time step $y_{\text{top}} - 3$. Now notice that there is no nice coordination of the choice and blocking meetings such that $c_{v_i,\gamma}$ can be drawn without a wiggle between $y_{\text{top}} - 3$ and $y_{\text{top}} - 1$. ◀

[1] That is also partially why we need x-coordinates being multiples of eight.

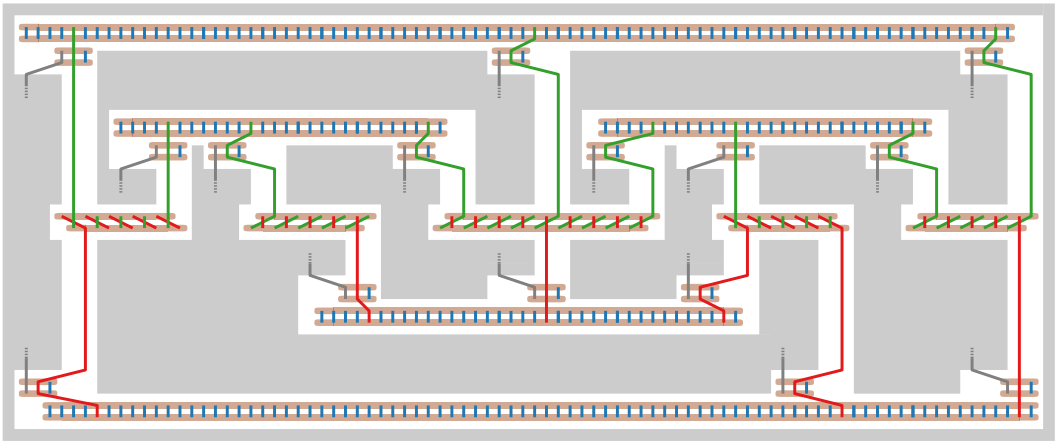


Figure 6 The complete reduction (with black-box frame) resulting from the instance in Figure 3.

▶ **Lemma 9.** *Assume a nice coordination in which some of $c_{v_i,\gamma}$ with $i \in [3]$ is satisfying. Then a nice coordination of the clause gadget has at least five wiggles, and such a coordination with exactly five wiggles always exists.*

Proof. For the existence, let $c_{v_j,\gamma}$ with $j \in [3]$ be some satisfying wire character. Simply draw $c_{v_j,\gamma}$ without wiggle, by adjusting the coordination of the choice meetings accordingly. The remaining two wire characters can be drawn with two wiggles, while their fixing character remains without wiggle between y_{bot} and y_{top} (see Figure 5a, where $j = 2$).

For the lower bound we have already seen in the proof of Lemma 8 that if a fixing character remains without wiggle between y_{bot} and y_{top}, then its corresponding wire character must have at least one wiggle between $y_{\text{top}} - 2$ and $y_{\text{top}} - 1$. Thus, for there to be at most 4 wiggles, two wire characters would need to be realized without wiggles. This is not possible because of the slight shift of positions of the wire characters in the choice meetings, a contradiction. ◀

We define k_{cr} as the number of crossings in the reduced storyline instance; in any coordination they will introduce a wiggle. Now, let $k = k_{\text{cr}} + 5|\Gamma|$. We claim the following; a proof without treating the rigid frame as black-box is given in the full version [7].

▶ **Lemma 10** (⋆). *I_{SAT} is a yes-instance of* Planar Monotone 3-Sat *if and only if I_{SL} has a nice coordination with at most k wiggles.*

Proof (assuming a black-box bounding frame). The forward direction follows immediately from our construction.

For the backward direction, assume a nice coordination of I_{SL} with at most k wiggles. Because k_{cr} wiggles are unavoidable, and we have at least 5 wiggles for each clause gadget by Lemmas 8 and 9, we must have exactly 5 wiggles per clause gadget. Further, each variable gadget has $|C_v|$ wiggles for each variable v. Hence, at least one wire character per clause gadget is satisfying, and we can use the variable assignment corresponding to satisfying characters to verify the satisfiability of the PM3-Sat instance by Observation 7. ◀

Proof of Theorem 5. For NP-membership we provide a polynomial certificate and a certifier for the decision problem of deciding whether there is a nice coordination with at most k wiggles. The certificate consists of a set of pairs $X \subseteq \mathcal{C} \times [\ell - 1]$ of size at least $\sum_{c \in \mathcal{C}}(|A(c)| - 1) - k$. Each pair $(c, t) \in X$ defines that c can be drawn without wiggle between time steps t and $t + 1$. The certifier can certify the certificate using a linear program. The linear program

is essentially the one from Section 3 using constraints (Cons-Δ) to (G0) and the additional constraints $y_{c,t} = y_{c,t+1}$ for each pair $(c,t) \in X$. If the polytope defined by the program is non-empty, then the certifier reports yes.

NP-hardness follows from Lemma 10 and the fact that PM3-Sat is NP-hard. Clearly, the reduction is polynomial. ◀

4.3 ILP Formulation for WCMin

We have established that WCMin is NP-complete. We now present an ILP formulation for WCMin (assuming $\Delta, \overline{\Delta} \in \mathbb{N}$) that we will use in Section 6 to construct storyline drawings with minimum wiggle count. The formulation requires the following lemma.

▶ **Lemma 11.** *Consider an ordered storylines instance and $\Delta, \overline{\Delta} \in \mathbb{N}$. For each $k \in \mathbb{N}$, if there is a nice coordination with k wiggles, there is one with at most k wiggles using integer coordinates.*

Proof. The proof is similar to the one of Proposition 1. We recommend reading that proof first. Consider a nice coordination with k wiggles. We build a graph G with $V(G) = \{y_{t,c} \mid t \in [\ell], c \in \mathrm{AC}(t)\}$ and $E(G) = \{y_{t,c}y_{t',c'} \mid y_{t,c} - y_{t',c'} \in \mathbb{Z}\}$. If G has multiple connected components, the y-coordinates corresponding to vertices in a single connected component can be increased until the number of connected components of G is decreased. When G is connected, all y-coordinates can be increased until they are all integers. ◀

The ILP makes use of the following variables (refer to Section 3 for details).

- $y_{t,c}$ for $t \in [\ell], c \in \mathrm{AC}(t)$, y-coordinate of character c at time step t.
- $z_{t,c}$ for $t \in [\ell-1], c \in \mathrm{AC}(t) \cap \mathrm{AC}(t+1)$, equals 1 if character c wiggles between t and $t+1$, 0 otherwise.

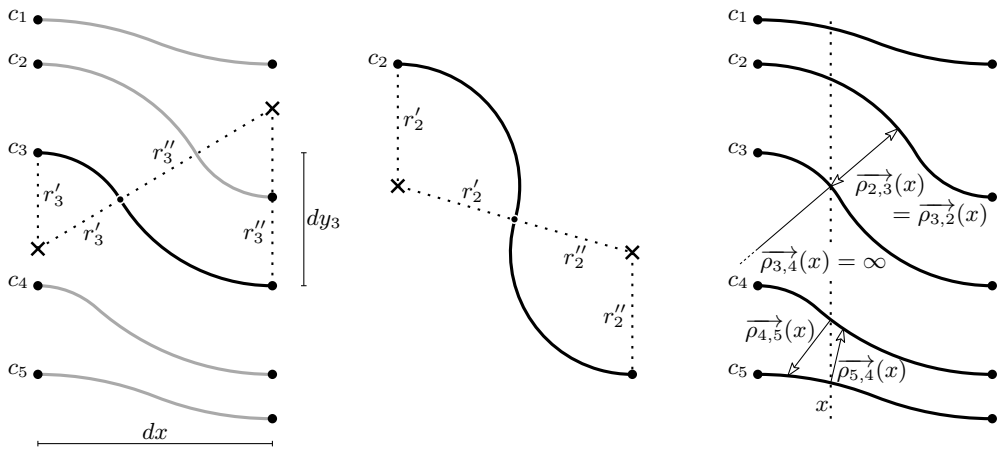
By Lemma 11 we can assume the y-coordinates of the characters to be integers. To enforce that $z_{t,c} = 0$ if and only if $y_{t,c} = y_{t+1,c}$ (i.e., there is no wiggle), we use a large constant $Y \in \mathbb{N}$; see constraints (M1) and (M2) below. A trivial upper bound for the y-coordinates is $\max\{\Delta, \overline{\Delta}\} \cdot \sum_{t \in [\ell]} |\mathrm{AC}(t)|$. We use this bound for Y. The sets $N_{\mathcal{M}}(t)$ and $N_A(t)$ are defined in Section 3. Now we can formulate the ILP:

$$\text{Minimize} \quad \sum_{t \in [\ell-1], c \in \mathrm{AC}(t)} z_{t,c}$$

$$\begin{aligned}
\text{subject to} \quad y_{t,c'} - y_{t,c} &= \Delta && \text{for all } t \in [\ell], (c,c') \in N_{\mathcal{M}}(t) && \text{(Cons-}\Delta\text{)}\\
y_{t,c'} - y_{t,c} &\geq \overline{\Delta} && \text{for all } t \in [\ell], (c,c') \in N_A(t) && \text{(Cons-}\overline{\Delta}\text{)}\\
y_{t,c} + Y \cdot z_{t,c} &\geq y_{t+1,c} && \text{for all } t \in [\ell-1], c \in \mathrm{AC}(t) \cap \mathrm{AC}(t+1) && \text{(M1)}\\
y_{t,c} - Y \cdot z_{t,c} &\leq y_{t+1,c} && \text{for all } t \in [\ell-1], c \in \mathrm{AC}(t) \cap \mathrm{AC}(t+1) && \text{(M2)}\\
y_{t,c} \in \mathbb{N}_0, \quad 0 \leq y_{t,c} &< Y && \text{for all } t \in [\ell], c \in \mathrm{AC}(t) && \text{(Int)}\\
z_{t,c} &\in \{0,1\} && \text{for all } t \in [\ell], c \in \mathrm{AC}(t) && \text{(Bin)}
\end{aligned}$$

Constraints (Cons-Δ) and (Cons-$\overline{\Delta}$) have been explained in Section 3. (Int) and (Bin) ensure that $y_{t,c}$ is a positive integer bounded by Y and $z_{t,c}$ is binary, respectively. The objective sums over all z-variables and therefore counts the number of wiggles.

As it turned out, optimal solutions created by state-of-the-art solvers produce unusably large wiggles with this formulation. We thus introduce a new variable h for the height of the drawing and, for each $t \in [\ell]$ and $c \in \mathrm{AC}(t)$, the new constraint $y_{t,c} \leq h$. By adding the term h/Y to the objective, we also minimize h. The adjusted formulation still solves WCMin.



(a) Wiggle geometry. **(b)** Not x-monotone. **(c)** Radial curve distance.

Figure 7 A wiggle is represented by a sequence of two circular arcs.

5 Routing Character Curves

In our storyline drawings, we represent each wiggle by a sequence of two circular arcs. In order to generate a smooth transition, we require the tangents at the arcs to be equal where the arcs cross over (see Figure 7a). In order to achieve visually pleasing transitions, we require that all arcs have at least a certain radius and that consecutive lines wiggle " in parallel" (a precise definition is given later in this section).

When visualizing the transition between a pair of consecutive time steps t and $t+1$, for each character c, the wiggle height $dy_{t,c} = |y_{t,c} - y_{t+1,c}|$ is given by the coordination. Given the radii $r'_{t,c}$ and $r''_{t,c}$ of the two arcs of a character curve c between time steps t and $t+1$, the required horizontal space dx_t for this configuration fulfills

$$dx_t^2 = 2(r'_{t,c} + r''_{t,c})dy_{t,c} - dy_{t,c}^2. \tag{4}$$

Refer to the full version [7] for a detailed derivation. We require each character curve to be x-monotone. If we choose dx_t badly, we may end up with a sequence of arcs that is not x-monotone (see Figure 7b). It is easy to see that a character curve is x-monotone if and only if $(r'_{t,c} + r''_{t,c}) \geq dy_{t,c}$. With Equation (4), we can state this as $dx_t^2 \geq dy_{t,c}^2$.

Between two time steps, two character curves run in parallel, if they do not cross and start and end with the same vertical distance. In order to have them wiggle " in parallel" , we use concentric arcs. Generally, for two x-monotone differentiable curves φ and φ', we define their *directed radial distance* at position x, denoted by $\overrightarrow{\rho_{\varphi,\varphi'}}(x)$, as the length of a normal line segment starting at $\varphi(x)$ and ending where it intersects φ' or infinity, if the normal line does not intersect φ'. See Figure 7c for an example. We define the *radial distance* of φ and φ' at x as $\rho_{\varphi,\varphi'}(x) = \min(\overrightarrow{\rho_{\varphi,\varphi'}}(x), \overrightarrow{\rho_{\varphi',\varphi}}(x))$ (see Figure 7c).

We call two x-monotone curves *related* if their axis-aligned bounding boxes overlap. We call them *upwards (downwards) co-oriented* if they are also y-monotone and their y-coordinates both increase (decrease). Let φ and φ' be two character curves between consecutive time steps t and $t+1$. By construction, φ and φ' are y-monotone. If they are also related, co-oriented, and do not cross each other, we require their radial distance to be monotone.

We define the set $N_{\uparrow,\leftarrow}(t)$ of all pairs of characters $c, \hat{c} \in \mathrm{AC}(t) \cap \mathrm{AC}(t+1)$ such that

- $y_{t,c} < y_{t,\hat{c}}$,
- c and $\hat{c}$ are non-crossing, related, and upwards co-oriented, and
- $y_{t,\hat{c}} - y_{t,c} \leq y_{t+1,\hat{c}} - y_{t+1,c}$.

We define $N_{\downarrow,\leftarrow}(t)$ for downwards co-oriented characters similarly as well as $N_{\uparrow,\rightarrow}(t)$ and $N_{\downarrow,\rightarrow}(t)$ if $y_{t,\hat{c}} - y_{t,c} > y_{t+1,\hat{c}} - y_{t+1,c}$. Note that these sets contain only characters that wiggle (i.e. their y-coordinates are different in t and $t+1$). We use the following LP to determine dx_t^2 as well as the radii $r'_{t,c}$ and $r''_{t,c}$ for each time step $t \in [\ell - 1]$ and for each $c \in \mathrm{AC}(t) \cap \mathrm{AC}(t+1)$.

$$\begin{aligned}
\text{Minimize} \quad & dx_t^2 \\
\text{subject to} \quad & dx_t^2 \geq \max_{c \in \mathrm{AC}(t)} dy_{t,c}^2 && && \text{(XM)} \\
& dx_t^2 = 2(r'_{t,c} + r''_{t,c}) dy_{t,c} - dy_{t,c}^2 && \text{for all } c \in \mathrm{AC}(t) \cap \mathrm{AC}(t+1) && \text{(R1)} \\
& r'_{t,c}, r''_{t,c} \geq r_{\min} && \text{for all } c \in \mathrm{AC}(t) \cap \mathrm{AC}(t+1) && \text{(R2)} \\
& r'_{t,c} \geq r'_{t,\hat{c}} + (y_{t,c} - y_{t,\hat{c}}) && \text{for all } c, \hat{c} \in N_{\downarrow,\leftarrow}(t) && \text{(D1)} \\
& r''_{t,c} \leq r''_{t,\hat{c}} - (y_{t+1,c} - y_{t+1,\hat{c}}) && \text{for all } c, \hat{c} \in N_{\downarrow,\rightarrow}(t) && \text{(D2)} \\
& r'_{t,c} \leq r'_{t,\hat{c}} - (y_{t,c} - y_{t,\hat{c}}) && \text{for all } c, \hat{c} \in N_{\uparrow,\leftarrow}(t) && \text{(D3)} \\
& r''_{t,c} \geq r''_{t,\hat{c}} + (y_{t+1,c} - y_{t+1,\hat{c}}) && \text{for all } c, \hat{c} \in N_{\uparrow,\rightarrow}(t) && \text{(D4)}
\end{aligned}$$

The constraint (XM) ensures x-monotonicity between time steps t and $t+1$; (R1) ensures that the two arcs meet at a distinct point and their tangents match. The remaining quality criteria are addressed by (R2) (minimum radius) and (D1) to (D4) (monotone radial distance). Note that we can model each variable of kind dx_t^2 as a linear variable because we do not have any terms depend on dx_t (non-squared).

6 Case Studies

We conducted two case studies that showcase wiggle minimization and the geometric realization we proposed. Firstly, we compare the different metrics for storyline wiggle (see Section 2) using both previous datasets from the literature and new instances. Secondly, we present a new use case for storylines when visualizing rolling stock schedules.

6.1 Storyline Benchmark Data

We compiled a diverse set of benchmark instances for which we could create crossing minimal ordered storylines using the ILP from [8]. It consists of three novels (n_1 to n_3) originally from The Stanford GraphBase [18] in a modified version by [13], eight blockbuster movies (b_1 to b_5 from [8] and b_6 to b_8 from [25]), one instance of publication data (p_1) from [9], and five rolling stock schedules (t_1 to t_5; see Section 6.2). Refer to a table in the full version [7] for more details about these instances, including objective values and running times.

We implemented the LP solving LWHMin (see Section 3.1), the QP solving QWHMin (see Section 3.2), and the ILP for WCMin (see Section 4.3) including the additional secondary objective that minimizes drawing height. We refer to them as LP, QP, and ILP, respectively. We implemented base – a heuristic that simply centers the character curves vertically in each time step – as a baseline. Note that base produces drawings of minimum height.
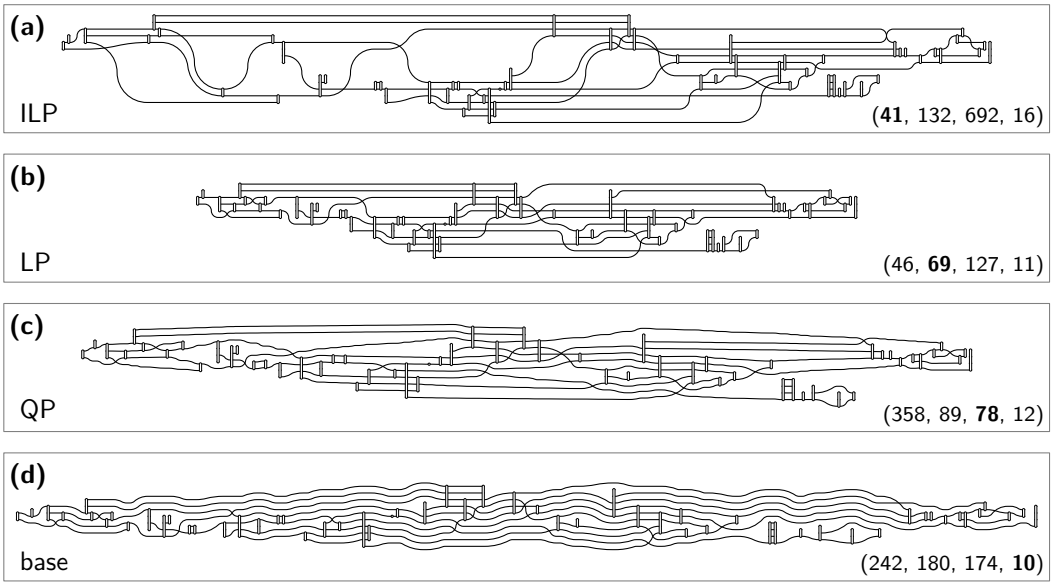


Figure 8 Storyline visualizations of the first chapter of Anna Karenina. The values in parentheses are (wiggle count, linear wiggle height, quadratic wiggle height, height of the bounding box).

Drawings of n_1 created with each of the four methods can be found in Figure 8. We see that minimizing the number of wiggles comes at a cost of some large wiggles and that producing many – thus small – wiggles results in an unnecessarily wide drawing when using our style. There could be a case for QP with an adjusted drawing style but for our current rendering, LP produces the best results.

Detailed results of the performance metrics for the dataset can be found in Figure 9. We see that optimizing for one objective can lead to poor performance in other metrics. Overall, however, LP produces good results more consistently than the other methods. This is why we used it for the drawings presented in our second case study.

6.2 Rolling Stock Scheduling

To our own surprise, storylines show up in a seemingly unrelated area, namely the visualization of rolling stock schedules. Rolling stock scheduling, which is an important subtask in railway optimization, is defined as follows. Given a set of train units, a set of possible compositions of train units, a set of trips given by the timetable, and passenger demands per trip, the task is to find a feasible assignment of train units (possibly in composition) to trips that minimizes a weighted sum of operating costs. These costs include, e.g., the total distance driven, the load factor, and the number of coupling operations. There are different approaches for visualizing rolling stock schedules (see [3] and [12, Fig. 1] for examples).

We visualize such a schedule as a storyline where each character curve represents the movement of a specific train unit over the planning period (usually one day of operation) and the composition of (typically two or three) train units is modeled as a meeting. In contrast to our basic model, such meetings now extend over several layers. Note that we use the term layer to avoid confusion with time steps in the timetable. In order to satisfy the additional constraints of "rolling stock" storylines, we needed to extend our basic model. We split each trip into at least two meetings, one at the departure time and one at the arrival time. We then group as many consecutive meetings (of pairwise different train units) into one layer as possible so the storyline stays compact. If departure and arrival meeting of one trip do not fall into consecutive layers, we insert an additional meeting for each intermediate layer.

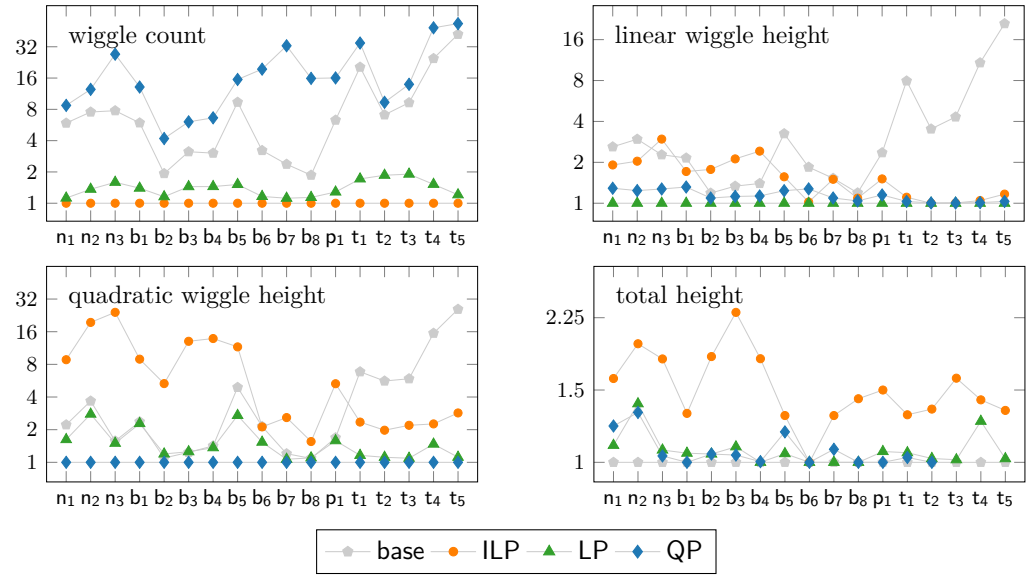


Figure 9 Metrics (relative to the respective optimum) of the wiggle minimization methods for our storyline benchmark set (**n**ovels, **b**lockbusters, **p**ublications, **t**rains). The y-axes are logarithmic.

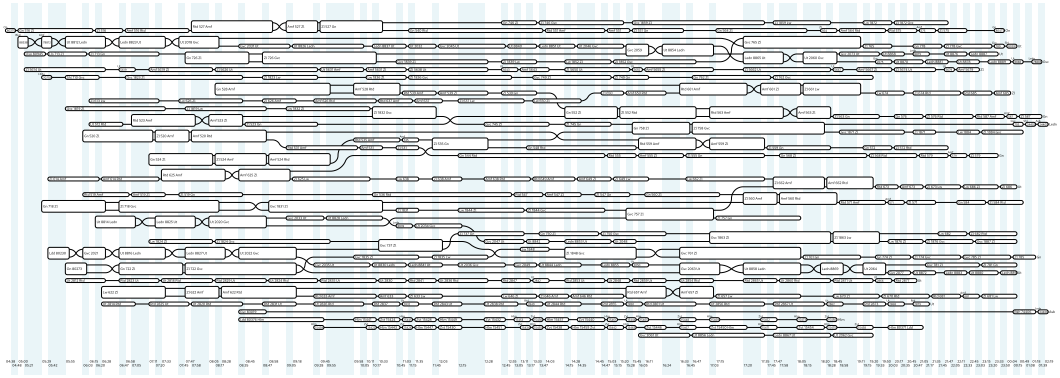


Figure 10 Storyline visualization of a rolling stock schedule using our drawing style.

Furthermore, we had to slightly adjust the ILP for crossing minimizing. We fixed the ordering of the characters within a meeting as they resemble the order of train units in a composition, which is relevant for planning. Additionally, we could have required that meetings belonging to the same trip must be drawn straight (as rectangles instead of as sequences of vertical bars) and that meetings cannot take part in a crossing, but for simplicity, we have not enforced this. However, we rewarded such solutions in the LP for linear wiggle height minimization using soft constraints. In the same way, we rewarded the vertical alignment of meetings where a train composed of the same units operates another trip.

An example of our visualization is shown in Figure 10 (larger versions are in the full version [7]). It depicts the same data set as [12, Fig. 1] (but unfortunately a different solution). Because of how we assign meetings to layers, the x-coordinates represent the departure and arrival times only vaguely. Also the width of a box does not represent the duration of a trip exactly. This was a deliberate (but optional) choice to achieve a more compact visualization.

7 Conclusion

We have given exact solutions for three variants of wiggle minimization. In particular, our efficient LP solution of LWHMin noticeably improves the aesthetic quality, while it does not cost much in terms of computation time (compared to crossing minimization).

We leave several questions open.

- We have shown that WCMin is NP-complete, but we can solve the restriction to two time steps efficiently. Is WCMin in FPT with respect to the number of time steps?
- Our geometric routing produces very readable drawings, but sometimes they are quite wide. Is there a more compact geometric solution that maintains the advantages of our method?
- Despite the rich literature on optimizing quality metrics in storyline visualization, there exists no formal user evaluation on the effects on readability of their optimization. In particular, do visualizations of rolling stock schedules as crossing- and wiggle-optimized storyline visualizations lead to improvements in train operation planning?

References

1 Dustin Arendt and Meg Pirrung. The "y" of it matters, even for storyline visualization. In Brian D. Fisher, Shixia Liu, and Tobias Schreck, editors, *Proc. Conference on Visual Analytics, Science and Technology (VAST 2017)*, pages 81–91, 2017. doi:10.1109/VAST.2017.8585487.

2 Marco Di Bartolomeo and Yifan Hu. There is more to streamgraphs than movies: Better aesthetics via ordering and lassoing. *Comput. Graph. Forum*, 35(3):341–350, 2016. doi:10.1111/CGF.12910.

3 Ralf Borndörfer, Boris Grimm, Markus Reuther, and Thomas Schlechte. Optimization of handouts for rolling stock rotations. *J. Rail Transport Planning & Management*, 10:1–8, 2019. doi:10.1016/j.jrtpm.2019.02.001.

4 Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 7th edition, 2009.

5 Lee Byron and Martin Wattenberg. Stacked graphs – geometry & aesthetics. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1245–1252, 2008. doi:10.1109/TVCG.2008.166.

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

7 Alexander Dobler, Tim Hegemann, Martin Nöllenburg, and Alexander Wolff. Optimizing wiggles in storylines. *CoRR*, abs/2508.19802, 2025. arXiv:2508.19802.

8 Alexander Dobler, Michael Jünger, Paul J. Jünger, Julian Meffert, Petra Mutzel, and Martin Nöllenburg. Revisiting ILP models for exact crossing minimization in storyline drawings. In Stefan Felsner and Karsten Klein, editors, *Proc. Graph Drawing and Network Visualization (GD 2024)*, volume 320 of *LIPIcs*, pages 31:1–31:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.GD.2024.31.

9 Alexander Dobler, Martin Nöllenburg, Daniel Stojanovic, Anaïs Villedieu, and Jules Wulms. Crossing minimization in time interval storylines. In Clemens Huemer and Carlos Seara, editors, *Proc. European Workshop on Compututational Geometry (EuroCG 2023)*, pages 36:1–36:7, 2023. doi:10.48550/arXiv.2302.14213.

10 Theresa Fröschl. Minimizing wiggles in storyline visualizations. Master's thesis, Technische Universität Wien, 2018. doi:10.34726/hss.2018.53581.

11 Theresa Fröschl and Martin Nöllenburg. Minimzing wiggles in storyline visualizations. In Fabrizio Frati and Kwan-Liu Ma, editors, *Proc. Graph Drawing and Network Visualization (GD 2017)*, volume 10692 of *Lecture Notes in Computer Science*, pages 585–587. Springer, 2018. URL: https://www.ac.tuwien.ac.at/files/pub/fn-mwsv-18.pdf.

12 Boris Grimm, Rowan Hoogervorst, and Ralf Borndörfer. A comparison of two models for rolling stock scheduling. *Transport. Sci.*, 59:1101–1129, 2025. doi:10.1287/trsc.2024.0505.

13 Martin Gronemann, Michael Jünger, Frauke Liers, and Francesco Mambelli. Crossing minimization in storyline visualization. In Yifan Hu and Martin Nöllenburg, editors, *Proc. Graph Drawing and Network Visualization (GD 2016)*, volume 8901 of *Lecture Notes in Computer Science*, pages 367–381. Springer, 2016. doi:10.1007/978-3-319-50106-2_29.

14 Tim Hegemann. NarratiViz. Software, BMFTR grant 01IS22012C, swhId: swh:1:dir:a7efa9c0ca674b6cf25b1c4b2c1f4793cc814723 (visited on 2025-11-10). URL: https://github.com/hegetim/narrativiz, doi:10.4230/artifacts.25067.

15 Tim Hegemann and Alexander Wolff. Storylines with a protagonist. In Stefan Felsner and Karsten Klein, editors, *Proc. Graph Drawing and Network Visualization (GD 2024)*, volume 320 of *LIPIcs*, pages 26:1–26:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.GD.2024.26.

16 Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Comb.*, 4(4):373–396, 1984. doi:10.1007/BF02579150.

17 Nam Wook Kim, Stuart K. Card, and Jeffrey Heer. Tracing genealogical data with timenets. In Giuseppe Santucci, editor, *Advanced Visual Interfaces (AVI)*, pages 241–248. ACM Press, 2010. doi:10.1145/1842993.1843035.

18 Donald E. Knuth. *The Stanford GraphBase: A platform for combinatorial computing*. ACM Press, 1994.

19 Irina Kostitsyna, Martin Nöllenburg, Valentin Polishchuk, André Schulz, and Darren Strash. On minimizing crossings in storyline visualizations. In Emilio Di Giacomo and Anna Lubiw, editors, *Proc. Graph Drawing and Network Visualization (GD 2015)*, volume 9411 of *Lecture Notes in Computer Science*, pages 192–198. Springer, 2015. doi:10.1007/978-3-319-27261-0_16.

20 Shixia Liu, Yingcai Wu, Enxun Wei, Mengchen Liu, and Yang Liu. Storyflow: Tracking the evolution of stories. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2436–2445, 2013. doi:10.1109/TVCG.2013.196.

21 Steffen Strunge Mathiesen and Hans-Jörg Schulz. Aesthetics and ordering in stacked area charts. In Amrita Basu, Gem Stapleton, Sven Linker, Catherine Legg, Emmanuel Manalo, and Petrucio Viana, editors, *Proc. Diagrammatic Representation and Inference (Diagrams 2021)*, volume 12909 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2021. doi:10.1007/978-3-030-86062-2_1.

22 Randall Munroe. Movie narrative charts. Diagram available at https://xkcd.com/657/, 2009. Accessed 2017/04/03.

23 Michael Ogawa and Kwan-Liu Ma. Software evolution storylines. In Alexandru C. Telea, Carsten Görg, and Steven P. Reiss, editors, *Proc. ACM Symposium on Software Visualization (SoftVis 2010)*, pages 35–42, 2010. doi:10.1145/1879211.1879219.

24 Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Syst. Man Cybern.*, 11(2):109–125, 1981. doi:10.1109/TSMC.1981.4308636.

25 Yuzuru Tanahashi, Chien-Hsin Hsueh, and Kwan-Liu Ma. An efficient framework for generating storyline visualizations from streaming data. *IEEE Trans. Vis. Comput. Graph.*, 21(6):730–742, 2015. doi:10.1109/TVCG.2015.2392771.

26 Yuzuru Tanahashi and Kwan-Liu Ma. Design considerations for optimizing storyline visualizations. *IEEE Trans. Vis. Comput. Graph.*, 18(12):2679–2688, 2012. doi:10.1109/TVCG.2012.212.

27 Thomas C. van Dijk, Martin Fink, Norbert Fischer, Fabian Lipp, Peter Markfelder, Alexander Ravsky, Subhash Suri, and Alexander Wolff. Block crossings in storyline visualizations. *J. Graph Alg. Appl.*, 21(5):873–913, 2017. doi:10.7155/jgaa.00443.

28 Thomas C. van Dijk, Fabian Lipp, Peter Markfelder, and Alexander Wolff. Computing storylines with few block crossings. In Fabrizio Frati and Kwan-Liu Ma, editors, *Proc. Graph Drawing and Network Visualization (GD 2018)*, volume 10692 of *Lecture Notes in Computer Science*, pages 365–378. Springer, 2018. doi:10.1007/978-3-319-73915-1_29.

An Algorithm for Accurate and Simple-Looking Metaphorical Maps

Eleni Katsanou
National Technical University of Athens, Greece

Tamara Mchedlidze
Utrecht University, The Netherlands

Antonios Symvonis
National Technical University of Athens, Greece

Thanos Tolias
National Technical University of Athens, Greece

Abstract

Metaphorical maps or *contact representations* are visual representations of vertex-weighted graphs that rely on the geographic map metaphor. The vertices are represented by countries, the weights by the areas of the countries, and the edges by contacts/boundaries among them. The *accuracy* with which the weights are mapped to areas and the *simplicity* of the polygons representing the countries are the two classical optimization goals for metaphorical maps. Mchedlidze & Schnorr [17] presented a force-based algorithm that creates metaphorical maps that balance between these two optimization goals. Their maps look visually simple, but the accuracy of the maps is far from optimal – the countries' areas can vary up to 30% compared to required. In this paper, we provide a multi-fold extension of the algorithm in [17]. More specifically:

1. Towards improving accuracy: We introduce the notion of region stiffness and suggest a technique for varying the stiffness based on the current pressure of map regions.
2. Towards maintaining simplicity: We introduce a weight coefficient to the pressure force exerted on each polygon point based on whether the corresponding point appears along a narrow passage.
3. Towards generality: We cover, in contrast to [17], non-triangulated graphs. This is done by either generating points where more than three regions meet or by introducing holes in the metaphorical map.

We perform an extended experimental evaluation that, among other results, reveals that our algorithm is able to construct metaphorical maps with nearly perfect area accuracy with a little sacrifice in their simplicity.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Mathematics of computing → Graph theory; Theory of computation → Graph algorithms analysis

Keywords and phrases Metaphorical maps, contact representation, accuracy (cartographic error), simplicity (polygon complexity), force directed algorithm

Digital Object Identifier 10.4230/LIPIcs.GD.2025.40

Related Version *E. Katsanou, T. Mchedlidze, A. Symvonis, T. Tolias, "An algorithm for accurate and simple-looking metaphorical maps"*: `https://arxiv.org/abs/2508.19810` [15]

Supplementary Material *Software*: `https://github.com/ekatsanou/metaphorical-maps`
archived at `swh:1:dir:44545be35a671a4fbe0f0ef758f6e9ed470e1bc4`

1 Introduction

Metaphorical maps or map-like graph visualization is an alternative way to the popular node-link diagrams [23] and matrix representations [27] for representing graphs. In these visualizations vertices are represented by polygonal regions and edges by contacts among


© Eleni Katsanou, Tamara Mchedlidze, Antonios Symvonis, and Thanos Tolias;
licensed under Creative Commons License CC-BY 4.0



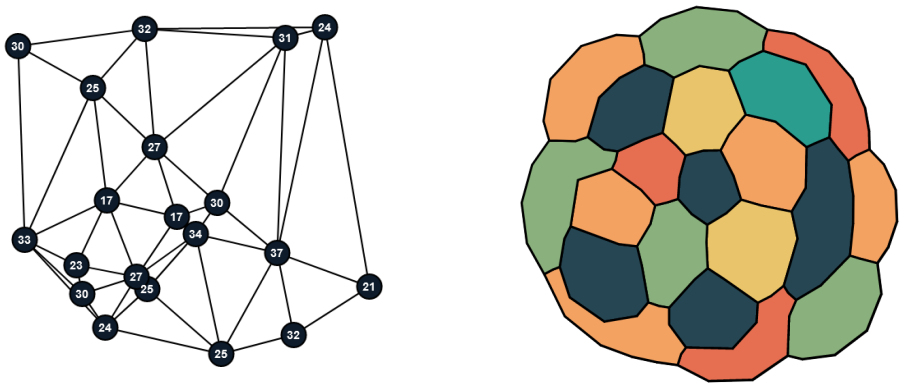


Figure 1 An internally triangulated plane graph with vertex weights and its corresponding metaphorical map.

them; see Fig. 1. Metaphorical maps have their individual advantage: they rely on the human familiarity with geographic maps and therefore instantly spark curiosity in the viewers by looking fairly familiar. It has been experimentally confirmed that map-like visualizations of graphs are more enjoyable than node-link diagrams [20]. They also outperform treemaps in the tasks that require recognition of hierarchy [6]. Another important advantage of these visualizations is that they can naturally display the weights associated with the graph's vertices, by the mean of resizing the map's regions. Such visualizations are also known as *area-proportional contact representations* [1] and are closely related to cartograms [25, 18], visualizations created by deforming geographic maps with a goal to represent values of a variable (e.g. population, number of votes) by the sizes of the deformed countries. Hogräfer, Heitzler, and Schulz [13] presented a unifying framework, showing that map-like representations of graphs and cartograms lie on two opposite sides of a single spectrum of map-like visual representations.

Map-like visual representations have been studied for more than 50 years [25] and a few excellent surveys have been devoted to this topic [13, 18, 25]. Generally, the construction of map-like representations is guided by the following criteria. *Statistical accuracy* refers to how well the modified areas represent the corresponding statistic shown; with the *cartographic error* measuring by how much the actual area of a region is away from the desired. *Geographical accuracy* refers to how much the modified shapes and locations of the regions resemble those in the original map. Preserving geographical accuracy is a goal in many algorithms generating cartograms. However, for metaphorical maps, the geographical accuracy is not relevant as there is no given geography to preserve. Finally, *topological accuracy* refers to how well the topology of the cartogram matches the topology of the original map. In case of map-like representations of planar graphs topological accuracy must be fully preserved – all edges of the given planar graph must be represented as regions' contacts and each region contact must correspond to an edge.

A criterion that is less explicitly mentioned in the map-like representation literature is the complexity of the outlines of map's regions. In general, keeping the region outlines simple is desired, since one of the tasks the users of these visualizations face, is the comparison of the region areas. The user performance in this task differs even when circles are compared to rectangles [10]. Motivated by this, a few works concentrated on generating map-like representations with simple regions. For instance, the Dorling cartogams [12] use circles as

regions, and Demers cartograms [7] use square regions. Also, mosaic cartograms [9] tend to have simpler looking outlines of regions, since they are constituted by segments having a limited amount of slopes. Contrary to that, the approaches where no care is taken of the complexity of the regions, tend to produce regions with very complex outlines, refer to [18].

In theory research, the complexity of regions was formalized as the maximum number of sides per region. A series of works was devoted to reducing this metric for rectilinear area-proportional contact representation from the initial 40 [11] to the final 8 [2]. Other relevant series of work on *area-universal planar drawings*, starts with a planar drawing and an assignment of weights to its faces. The requirement is to modify the drawing, by preserving its planar embedding and realize the desired face areas. Thomassen [24] showed that every plane cubic graph is *area-universal* (perfect cartographic accuracy is achievable) and additionally in the resulting map the faces are bounded by triangles. Kleist [16] showed that 1-subdivision of any plane graph is area-universal. However, all the theoretical approaches that guarantee a few sides per region and perfect cartographic accuracy [2, 16, 24] are not guaranteed from having very narrow pointed areas in the regions and therefore the diagrams do not really look simple; refer for instance to Fig. 12.(b,f) .

Motivated by this challenge, Mchedlidze and Schnorr [17] used a more sophisticated complexity measure introduced in [8] to evaluate the quality of the polygons in their metaphorical maps, that is able to capture the intuitive perception of a polygon's complexity. They presented a force-based algorithm that creates metaphorical maps that balance between the two optimization goals: cartographic error and polygon complexity. Their maps look visually simple, but the cartographic error can be up to 30%.

Our contribution. The known techniques either build complex metaphorical maps with perfect cartographic accuracy [2, 16, 24] or simple-looking metaphorical maps with cartographic error up to 30% [17]. In this work, we address this gap and aim towards producing simple metaphorical maps with near zero cartographical error.

- We present a multi-fold extension of the algorithm in [17]. More specifically:
  - Towards improving accuracy: We introduce the notion of region stiffness and suggest a technique for varying the stiffness based on the current pressure of map regions.
  - Towards maintaining simplicity: We introduce a weight coefficient to the pressure force exerted on each polygon point based on whether the corresponding point appears along a narrow passage.
  - Towards generality: We cover, in contrast to [17], non-triangulated graphs. This is done by either generating points where more than three regions meet or by introducing holes in the metaphorical map.
- We perform an extended experimental evaluation which aims to:
  - Compare our algorithm to the algorithm of Mchedlidze and Schnorr [17] with respect to cartographic accuracy and complexity.
  - Examine the cartographic accuracy vs the regions' complexity trade-off.
  - Evaluate the extension to non-triangulated graphs.
  - Evaluate the influence of the initial layouts on the quality of the maps, by comparing the initial layout as suggested in [17] to the cartographically accurate initial layout given by the theoretical approach in [3].
- Our experiments among other facts indicate that the suggested algorithm achieves almost perfect cartographic accuracy with only a small increase in polygon complexity compared to [17].

Paper organization. In Section 2, we present our notation, the quality metrics employed in evaluating cartographic maps as well as the basic ingredients of the algorithm of Mchedlidze and Schnorr [17]. In Section 3, we present our algorithm which (a) introduces the notion of region stiffness, (b) applies corrective weight coefficients on pressure forces, and (c) handles non-triangulated plane graphs. Our experimental evaluation is presented in Section 4. Finally, we conclude in Section 5 with directions for future work.

A demo application for generating metaphorical maps has been implemented in JavaScript using yFiles [28] and is available as a web application at `http://aarg.math.ntua.gr/demos/metaphoric_maps/`. All experiments were executed using a Java implementation. To allow replicability, we make the implementation and the whole graph benchmark publicly available at `https://github.com/ekatsanou/metaphorical-maps`.

2 Preliminaries

In this section, we introduce the notation used throughout the paper and we present the quality metrics of cartographic accuracy and polygon complexity that are employed in our experimental evaluation. Given that the algorithm presented in this paper is an extension of the algorithm of Mchedlidze and Schnorr [17] (shortly `MS-algorithm`), we also present its brief description.

Let $G = (V, E, w)$ be a vertex-weighted graph where $w : V \to \mathbb{R}^+$ is its weight function. In a *metaphorical map* $M = \mu(G)$ of G, vertex $v \in V$ is depicted as polygonal *region* (*country*) $\mu(v)$ so that adjacent vertices share a non-trivial contact (*boundary*). Our optimization goal is to build metaphorical maps such that the area of the region $\mu(v)$, denoted by $A(\mu(v))$, is roughly equal to $w(v)$.

In a metaphorical map $M(G)$, we refer to the points that define its polygonal regions also as *vertices*. Given two vertices u, v of the metaphorical map, we denote by $\mathbf{uv}$ the vector from u to v.

2.1 Quality measures for Metaphorical Maps

As mentioned in the introduction, we measure the quality of a metaphorical map by its cartographic accuracy – for this, similarly to the previous work, e.g. [2, 4, 17], we employ the metric *normalized cartographic error*. The simplicity of the map is measured by the metric *polygon complexity*, introduced in [8] and applied for the evaluation of metaphorical maps in [17].

Normalized cartographic error. Consider a vertex $v \in V$ of a vertex-weighted graph $G = (V, E, w)$ and its corresponding region $\mu(v)$ in its metaphorical map $\mu(G)$. The *normalized area of region* $\mu(v)$, denoted by $A'(\mu(v))$, is defined as:

$$A'(\mu(v)) \coloneqq A(\mu(v)) \cdot \frac{\sum_{u \in V} w(u)}{\sum_{u \in V} A(\mu(u))}$$

Then, the *normalized cartographic error* E_{nc} *of region* $\mu(v)$ is defined as:

$$E_{nc}(\mu(v)) \coloneqq \frac{\mid A'(\mu(v)) - w(v) \mid}{\max\{A'(\mu(v)), w(v)\}} \tag{1}$$

Polygon Complexity. Brinkhoff, Kriegel, Schneider, and Braun [8] defined the complexity of a polygon P as a function of three quantities (see also [15] for details), that can be intuitively understood based on an example of a star-shape: (a) the *frequency of P's vibration*, denoted by $\text{freq}(P)$, which specifies how many tips a star-shape has, (b) the *amplitude* of P's vibration, denoted by $\text{ampl}(P)$, which specifies how long the tips of a star-shape are and (c) P's *convexity*, denoted by $\text{conv}(P)$, the fraction of the area that is not covered within the smallest enclosing circle. The *polygon complexity* is then defined as $\text{compl}(P) = 0.8 \cdot \text{ampl}(P) \cdot \text{freq}(P) + 0.2 \cdot \text{conv}(P)$, which maps P to values in $[0, 1]$, with low values meaning low complexity.

In our experiments, we quantify the quality of a map by both the average and the maximum value of normalized cartographic error and polygon complexity over the regions of the map.

2.2 The `MS-algorithm`

The `MS-algorithm` [17] is a typical force-directed algorithm that employs several antagonistic forces applied to the vertices of the metaphorical map, that, hopefully, at equilibrium produce a layout with good characteristics, i.e., low normalized cartographic error and low polygon complexity. The algorithm employs three forces (vertex-vertex repulsion, vertex-edge repulsion, and angular-resolution) targeted towards producing metaphorical maps of low polygon complexity and one force (air-pressure) which works towards reducing the cartographic error of the metaphorical map. Here, we only describe the air-pressure force, since our proposed algorithm modifies how this force is applied. The exact definition of the other forces can be found in [15] and in [17].

The *normalized air pressure* in region g, $P(g)$, exerts a force on each bounding edge e based on the pressure's magnitude and the edge's length $\ell(e)$ in relation to the length of the entire polygonal region boundary $\text{circ}(g)$; see [15]for a formal definition of $P(g)$. Thus, the *air-pressure force* on edge e of region g was defined as $\mathbf{F}(e, g) = 3P(g)\frac{\ell(e)}{\text{circ}(g)}\widehat{\mathbf{r}}$, where $\widehat{\mathbf{r}}$ is a unit vector perpendicular to e directed towards outside of g. Force $\mathbf{F}(e, g)$ was applied to both endpoints of edge e.

The force-directed algorithm is run for *iter* number of steps, the value of this parameter is determined experimentally. Notice that the resultant of the forces applied to a vertex of the map, may force it to cross over an edge, and thus, the displacement of each vertex must be limited to prevent this from happening. To achieve that, the `MS-algorithm` adopted the rules of ImPrEd [22] that ensure the preservation of the planar embedding of the map. It should be also noted that the constant factors employed in each of the described forces have been determined experimentally.

A final note regarding the number of vertices defining each polygonal region. This number fluctuates during the execution of the algorithm in order to allow each region to obtain a more elaborate (or simpler) shape. Let $\bar{\ell}$ denote the average edge length over all edges in the metaphorical map. Provided that no edge crossings are introduced, the `MS-algorithm` eliminates vertices of degree 2 that become closer than $\frac{1}{10}\bar{\ell}$ to their neighbor and it splits in half the edges that get longer than $2\bar{\ell}$.

Each force-directed algorithm requires for its execution an initial layout. Given a plane vertex-weighted graph G, the `MS-algorithm` constructs an initial metaphorical map by considering the dual of a planar drawing of G. The algorithm is designed for internally triangulated graphs. The dual vertices of the inner faces are placed in the barycenters of the triangles representing those faces. The dual edges are drawn as polylines consisting of two segments meeting at a bend-point, which lies in the middle of the corresponding primal edge. Details can be found in [17] and in [15].

3 Our Metaphorical Map Generation Algorithm

In this section, we describe our extension of the `MS-algorithm`. More specifically, we describe how to revise the air-pressure force utilized in the `MS-algorithm` by (a) incorporating a *stiffness coefficient* and (b) by applying a corrective *weight coefficient* that aims to eliminate narrow passages in the metaphorical map. The constants involved in the calculations of our algorithm are determined experimentally. Finally, we show how the algorithm is adapted in order to accommodate non-triangulated graphs.

The Stiffness of each Map Region. After examining the output of the `MS-algorithm`, we observed that some regions had noticeably higher cartographic error than others, leading to the conclusion that some attributes of a region made it less responding to air pressure and, thus, resulted in high cartographic errors. Therefore, for every region we introduce a variable, which we refer to it as *stiffness coefficient*, that accounts for its stiffness. Let g be an internal region and let $P_i(g)$ be its pressure at the i-th iteration of the force-directed algorithm. If at iteration i region g is over-pressured (i.e., $P_i(g) > 1$) we increase the region's stiffness coefficient by a small amount *step*; if it is under-pressured (i.e., $P_i(g) < 1$) we decrease it by *step*, otherwise, it remains unchanged. In addition, we restrict the stiffness coefficient in the range $[s_{low}, s_{high}]$ where $s_{high} \geq 1$ and $s_{low} = \frac{1}{s_{high}}$. An appropriate value for s_{high} is determined experimentally.

Given a region g, we define the *stiffness coefficient of g at the i-th iteration of the algorithm*, denoted by $s_i(g)$, as:

$$\begin{cases} s_0(g) = 1 \\ s_i(g) = \min(s_{high}, \max(s_{low}, s_{i-1}(g) + \alpha \cdot step)), & i > 0 \end{cases}$$

where $\alpha = \begin{cases} -1, & P_{i-1}(g) < 1 \\ 0, & P_{i-1}(g) = 1 \\ 1, & P_{i-1}(g) > 1 \end{cases}$

Then, the *revised air-pressure force* of region g on its boundary edge e during the i-th iteration of the algorithm, $i \geq 1$, denoted by $\mathbf{F'_i}(e, g)$ is defined as

$$\mathbf{F'_i}(e, g) = s_i(g) \cdot \mathbf{F_i}(e, g) \tag{2}$$

where $\mathbf{F_i}(e, g)$ is the air-pressure force computed at the i-th iteration of the `MS-algorithm`. Note that the "stiffness" attribute is different for each region and demonstrates an adaptive behaviour over time. Employing such adaptive coefficients in force/energy-directed drawing algorithms appears to be uncommon, mainly due to performance issues. However, we note that a similar scheme has been employed in the work of Hu [14] (referred to as *adaptive cooling scheme*).

We performed a few experiments (see [15] for details) with 20-node graphs in order to determine the parameters of the algorithm, that resulted in the following values: $step = 0.02$, $iter = 1.000$ and $s_{high} = 8$. To also account for the size of the input graph, assuming that larger graphs take a longer time to converge to a good metaphorical map, we decided to set the number of iterations to $iter = 800 + 10n$, the value consistent with $iter = 1.000$ for $n = 20$.

Improving the Visual Complexity. When we applied the stiffness coefficients to the regions, we observed that they tended to create long, narrow passages, thereby reducing the visual complexity of the metaphorical map. To mitigate this effect, we introduce a new corrective

weight coefficient for the pressure forces exerted on each region's vertices, while keeping the total pressure of the region constant. These weight coefficients redistribute pressure so that vertices in narrow passages receive a larger share of the force, whereas vertices in wider parts of the region receive less. Before defining the weight coefficient, we introduce some auxiliary notation:

- Let M be a metaphorical map, and let $\rho(M) = \sqrt{\left(\sum_{g \in M} A(g)\right)/\pi}$ denote the radius of a circle whose area equals the total area of M. We refer to $\rho(M)$ as the *ideal radius* of M.
- Let g be a region of M, let u be one of its vertices and let e be an edge of g that is not adjacent to u. Further, let x be the point of e that is closest to u. The Euclidean distance of u to e, denoted by $d_E(u, e)$, is simply the Euclidean distance from u to point x. We also define the *polygonal distance* of vertex u to edge e, denoted by $d_P(u, e)$, as the length of the shortest path when moving from u to x on the boundary of g.

Intuitively, an edge e is sufficiently "opposite" to a vertex u if it is close to it and, at the same time, it belongs to the opposite side of a narrow passage. For each vertex u of region g, we want to identify the edge e of g that is the closest out of those opposite to it. In order to avoid selecting edges that are collinear with u, we consider only edges for which it holds that $d_E(u, e) < 0.9\, d_P(u, e)$ and, out of those, we select the edge e of smallest Euclidean distance to u. We refer to this edge e as the *pairing edge* of u and we denote it by $pair(u)$. Further, we denote the Euclidean distance from u to $pair(u)$ by $d(u, g)$.

Consider the metaphorical map M. We introduce a scale factor based on the ideal radius $\rho(M)$. We regard $0.05\,\rho(M)$ as the minimum "acceptable" passage width and, hence, we define the quantity $\delta(u, g) = \frac{0.05\,\rho(M)}{d(u,g)}$ which assumes values greater than 1 when u is very close to its pairing edge $pair(u)$.

We now define the corrective coefficient $\beta(u, g)$ that will be applied to the air-pressure forces at each vertex u of region g of the metaphorical map as:

$$\beta(u, g) = 1 + \operatorname{sign}\big(\delta(u, g) - 1\big) \ln\Big(1 + \big|\delta(u, g) - 1\big|\Big).$$

Hence:

- If $\delta(u, g) = 1$, i.e., u marginally does not participate in a narrow passage, then $\beta(u, g) = 1$.
- If $\delta(u, g) > 1$, then $\beta(u, g) = 1 + \ln\big(\delta(u, g)\big)$, which grows logarithmically once $\delta(u, g)$ exceeds 1.
- If $0 < \delta(u, g) < 1$, then $\beta(u, g) = 1 - \ln\big(2 - \delta(u, g)\big)$, which remains below 1 but varies smoothly as $\delta(u, g)$ decreases.

Let u_i and u_{i+1} be the endpoints of edge e_i of region g and let $\widehat{\mathbf{r}}$ be a unit vector perpendicular to e_i directed towards the exterior of g. Recall that, the air-pressure force on e_i is defined as $\mathbf{F}(e_i, g) = 3P(g)s(g)\frac{\ell(e_i)}{\mathrm{circ}(g)}\widehat{\mathbf{r}}$, where $s(g)$ is the stiffness coefficient in the current iteration. Since the air-pressure force is exerted on both endpoints of e_i, the total air-pressure force on e_i is $2 \times 3P(g)s(g)\frac{\ell(e_i)}{\mathrm{circ}(g)}\widehat{\mathbf{r}}$ and the total air-pressure force of region g is $2 \times 3P(g)s(g)$.

The new air-pressure force, employing the above introduced corrective coefficient $\beta(\cdot, g)$, on edge $e_i = (u_i, u_{i+1})$ is now defined as $\mathbf{F}(e_i, g) =$

$$\underbrace{3P(g)s(g)\frac{2\beta(u_i, g)\ell(e_i)}{\sum_j \ell(e_j)(\beta(u_j, g) + \beta(u_{j+1}, g))}\widehat{\mathbf{r}}}_{\text{Force applied on } u_i} + \underbrace{3P(g)s(g)\frac{2\beta(u_{i+1}, g)\ell(e_i)}{\sum_j \ell(e_j)(\beta(u_j, g) + \beta(u_{j+1}, g))}\widehat{\mathbf{r}}}_{\text{Force applied exerted on } u_{i+1}}.$$

Therefore, the total air-pressure force applied to region g equals to $\sum_i \mathbf{F}(e_i, g)$. So, we have that:

$$\sum_i \mathbf{F}(e_i, g) = \frac{6P(g)s(g)}{\sum_j \ell(e_j)(\beta(u_j, g) + \beta(u_{j+1}, g))} \sum_i (\beta(u_i, g)\ell(e_i) + \beta(u_{i+1}, g)\ell(e_i))$$

$= 2 \times 3P(g)s(g)$, and thus, the total air-pressure force on region g remains unchanged.

3.1 Dealing with non-Triangulated Graphs

In this section, we discuss how to create metaphorical maps for biconnected, non-triangulated plane graphs. Mchedlidze and Shnorr [17] focused on the case of internally triangulated graphs. However, their algorithm was, in principle, able to also deal with non-triangulated graphs provided that, in the initial layout of the given graph, the *barycenter visibility property* is satisfied, that is, the barycenter of each non-triangulated face is located within that face and all vertices of the face are *visible* from the barycenter, that is, the open segment that connects the barycenter and any vertex of the face does not cross the boundary of the face.

We proceed to establish the barycenter visibility property as follows. We internally triangulate the graph by creating a new *auxiliary* vertex for each non-triangulated inner face and connecting it to every vertex of the face. The use of auxiliary vertices for mesh generation, often called Steiner points, is a common technique to transform general grids into high-quality triangulations (see [5, 19]). Let $G' = (V \cup V_{aux}, E \cup E_{aux})$ be the resulting internally triangulated graph where V_{aux} and E_{aux} are the sets of auxiliary vertices and the extra edges used in the triangulation, respectively. We apply on G' Tutte's barycentric embedding algorithm [26] that fixes the outerface on a circle and places every inner vertex on the barycenter of its neighbors, resulting in a planar straight line layout $\Gamma_{bar}(G')$; refer to Fig. 2.(a-b) We finish by removing the auxiliary vertices and their adjacent edges from $\Gamma_{bar}(G')$; refer to Fig. 2.(c). It is trivial to see that the derived layout satisfies the barycenter visibility property. Indeed, the auxiliary vertex of each non-triangulated face f is placed at the barycenter of its neighbors, that is, the vertices of f. Therefore, in the constructed planar drawing, face f has its barycenter lying inside it. Moreover, the fact that the auxiliary edges inside f do not cross the boundary of f, ensures the visibility between the barycenter of f and all of f's vertices.

After ensuring that the barycenter of every inner face lies within its boundary, we are in a position to obtain an initial metaphorical map by using the transformation deployed in the `MS-algorithm`; refer to Section 2.2 and Fig. 2.(d). Note that this transformation leads to points in the metaphorical maps where more than three regions meet. Recall that in our definition of the metaphorical map, only the non-degenerate contacts represent adjacencies, therefore the maps constructed this way still represent exactly the adjacencies present in the given graph.

Metaphorical maps with holes. By visually inspecting the maps created with the above method, we noticed that the large faces (in the initial graph) which result in multiple point contacts (in the map) appear to be quite cluttered; refer to Fig. 2.(d) and Fig. 3.(a). Therefore, we propose an alternative method that treats large faces as holes; Refer to Fig. 2.(e). Practically, obtaining such maps is very simple – we can just consider the graph G' obtained after the introduction of the auxiliary vertices, transform it to a map, as suggested in the `MS-algorithm` , and treat regions corresponding to the auxiliary vertices as holes. When applying forces to the map vertices, we can treat holes in exactly the same way we treat other

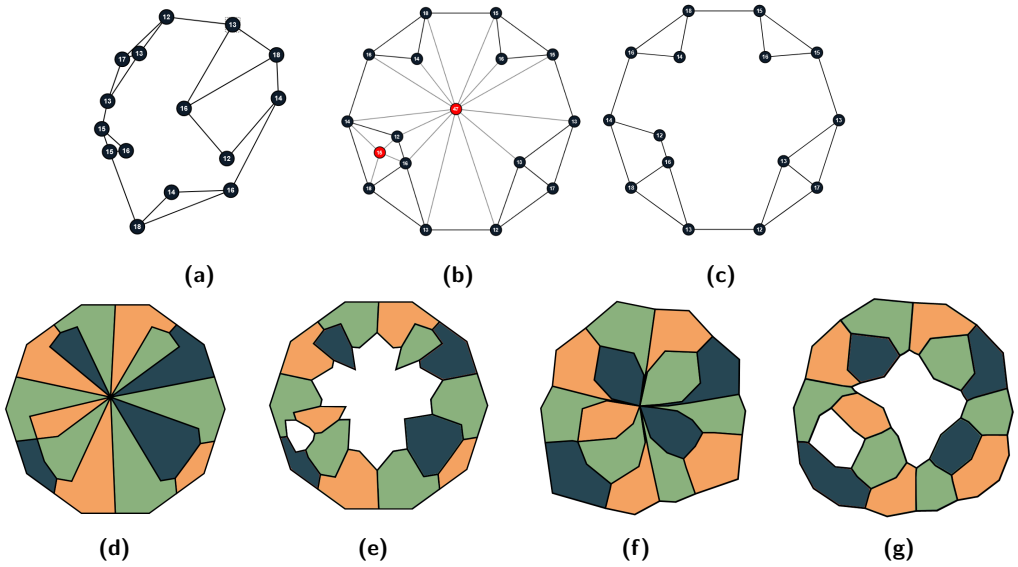


Figure 2 Construction of the initial layout for non-triangulated graphs. (a) A graph G with a large non-triangulated face. (b) The auxiliary graph G' – red vertices are added, drawn with the Tutte's algorithm. (c) The auxiliary vertices are removed. (d) Transformation of the drawing in Fig. 2.(c) based on the graph's dual (as employed in the `MS-algorithm`). (e) Transformation of the drawing in Fig. 2.(c) based on the graph's dual and by treating the auxiliary vertices as holes. (f-g) Forces are applied to Fig. 2.(d) and Fig. 2.(e), respectively; holes are weighted according to the proposed function.

regions. However, since the holes do not have a target weight, we have the flexibility to assign those weights with the goal to improve the appearance of the overall map. Our intuition is that the more regions are adjacent to a hole, and the heavier those regions are, the bigger the hole needs to be. After some experimentation, we found that the following function for assigning weights to holes produces reasonably good results, however, with further testing better functions may arise:

$$holeWeight(h_i) = \frac{1}{4\deg(h_i)} \left[\sum_{(v_j,h_i)\in E(G)}^{j} \sqrt{w(v_j)} \right]^2 \tag{3}$$

Note that, we do not prioritize minimizing the normalized cartographic error within holes, as they do not represent regions. However, it is important that holes remain visually simple to preserve the clarity of adjacencies. To allow adjacent regions the flexibility to expand naturally, we do not apply stiffness coefficients to holes. Nonetheless, we do apply the corrective weight coefficients of the pressure forces to maintain overall layout stability.

4 Experimental Evaluation

In our experimental evaluation we aim to answer the following research questions:

1. What is the performance of our algorithm compared to the performance of the `MS-algorithm` based on the quality metrics of normalized cartographic error and polygon complexity? Does the difference in performance depend on the input characteristics such as number of vertices, nesting ratio or weight ratio (refer to Section 4 for the definitions)?



(a) Map with high degree contact points. **(b)** Map with holes.

Figure 3 Two maps which correspond to the same 40-vertex graph. Non-triangulated regions are represented in (a) with high degree contact points and in (b) with holes.

2. Is there a trade-off between cartographic error and polygon complexity? How do we control it?
3. Does the initial metaphorical map have an effect on the quality of the final one?
4. Is the handling of non-triangulated biconnected graphs satisfactory?
5. Is the proposed algorithm practical? Does its running time scale well with the size of the input graph?

The analysis of question 5 can be found in [15].

The Graph Test Data. For the generation of test data, we followed the approach of [17] (described in detail in Algorithm 5.1 of [21] with a uniform distribution). The authors of [17] generate planar straight-line drawings based on the Delaunay triangulation of a random point set. A fraction $nest \in [0, 1]$ of the points, called *nesting ratio*, is placed within the triangles of the initial Delaunay triangulation. It was experimentally verified in [17] that the nesting ratio has an effect on the algorithm's performance, therefore, we also included it in our analysis. For every vertex, we generate a weight using a uniform distribution with different *weight ratios* of maximum to minimum weight. The above described procedure is augmented to generate non-triangulated plane graphs as follows. Following the generation of an internally triangulated plane graph, we remove a number of randomly selected internal edges in order to create non-triangulated plane graphs. We denote the ratio of removed to initial internal edges by *rem*. Note that an edge is only removed if the graph remains biconnected.

Comparison to the `MS-algorithm`. For the comparison of the performance of our algorithm (`NEW-algorithm`) against `MS-algorithm`, we examined how three parameters – the nesting ratio *nest* (see Fig. 4), the weight ratio w (see Fig. 5), and the total number of nodes n (see Fig. 6)–influence both algorithms. To this end, we conducted three experiments:

1. *Varying nesting ratio.* For each value of $nest \in \{0, 0.1, 0.2, \ldots, 1.0\}$, we generated 50 graphs with $n = 20$ nodes and weight ratio $w = 5$.
2. *Varying weight ratio.* For each $w \in \{5, 10, 15, 20\}$, we generated 50 graphs with $n = 20$ nodes and nesting ratio $nest = 0$.
3. *Varying number of nodes.* For each $n \in \{15, 20, 25, \ldots, 80\}$, we generated 50 graphs with nesting ratio $nest = 0$ and weight ratio $w = 5$ and we let the algorithms run for $800 + 10n$ iterations.

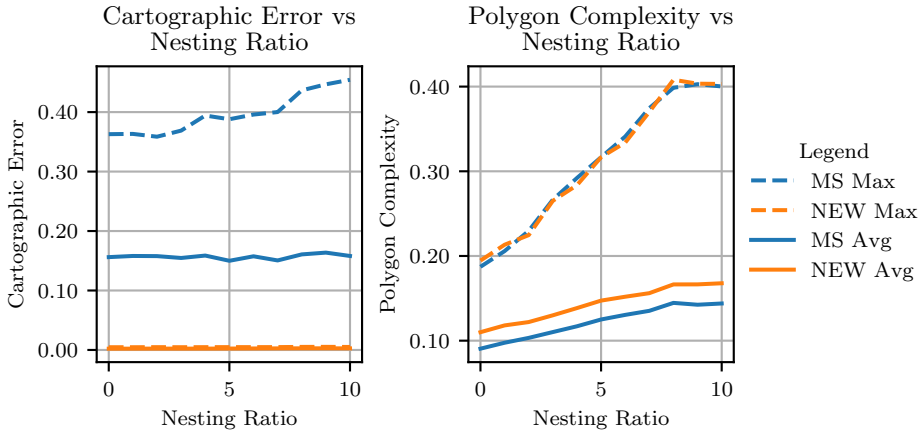


Figure 4 `NEW-algorithm` vs `MS-algorithm` for different values of nesting ratio.

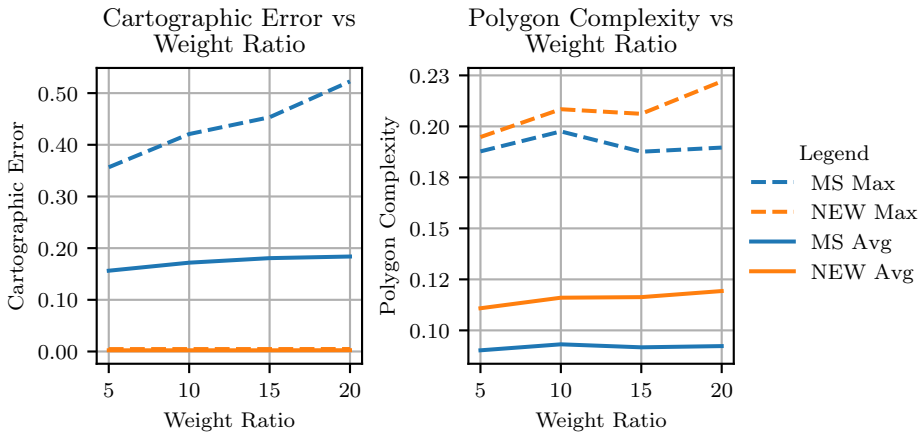


Figure 5 `NEW-algorithm` vs `MS-algorithm` for different values of weight ratio.

The results of every comparison follow a similar trend. Our algorithm achieves average cartographic error close to zero at the cost of a slightly higher polygon complexity. In all of our experiments (1450 in total), the recorded average normalized cartographic error (for each metaphorical map created) was in the range $[0.12\%, 0.44\%]$. In comparison, the same metric for `MS-algorithm` is in the range $[8.8\%, 24.7\%]$. The price we paid was a small increase in the average polygon complexity. The highest observed increase in average polygon complexity over all the maps was 6.9% and was observed for $n = 75$, where the maximum value of average polygon complexity was 19.6%, well below the 40% value where polygons are considered to be complex [8]. The maps in our experiment that correspond to this maximum value of increase in the average polygon complexity are shown in Fig. 7.(a-b). We observe that indeed our map's regions are longer and, sometimes, narrower. However, in the case of smaller (or "simpler") graphs, this increase in complexity is hardly noticeable, see e.g. Fig. 7.(c-d). Regarding the properties of the graphs, all three parameters (nesting ratio, weight ratio, and the number of vertices) affect the polygon complexity of our produced maps.

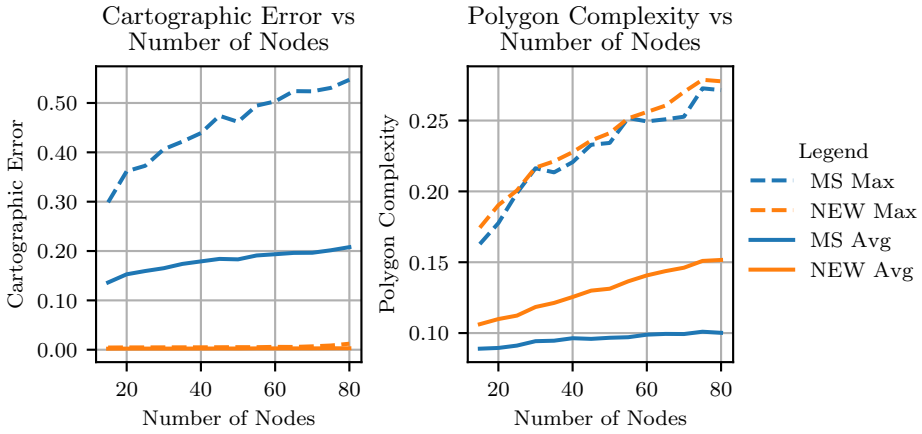
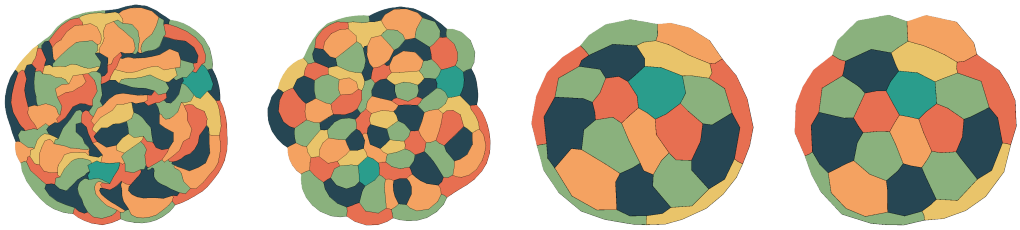


Figure 6 `NEW-algorithm` vs `MS-algorithm` for different number of nodes.



(a) `NEW-algorithm` **(b)** `MS-algorithm` **(c)** `NEW-algorithm` **(d)** `MS-algorithm`

Figure 7 (a-b) The maps that correspond to the largest increase in average polygon complexity. The corresponding graph has 75 vertices. (c-d) Maps for a random graph.

Cartographic error and polygon complexity trade-off. The cost we pay for achieving average cartographic error close to zero is a small increase in the average polygon complexity. Given that our algorithm is identical to the `MS-algorithm` for stiffness $s_{high} = 1$ (without the application of the corrective weight coefficients on the pressure forces), we evaluated our algorithm for several values of $s_{high} \in \{2, 4, 8\}$ (see Fig. 8). We observed that while the average and maximum normalized cartographic error remain nearly identical for $s_{high} = 4$ and $s_{high} = 8$, the algorithm performs better at $s_{high} = 8$ in terms of both average and maximum polygon complexity. Based on these observations, we set $s_{high} = 8$ in our algorithm. This value achieves very small cartographic error while maintaining acceptable polygon complexity. Moreover, increasing s_{high} beyond 8 does not lead to meaningful improvements in either evaluation metric.

Fig. 9 shows a "heat-map" style coloring of the regions of metaphorical maps of the same graph for `MS-algorithm` (Fig. 9.(a)) and our algorithm for $s_{high} \in \{2, 8\}$ (Fig. 9.(b)-(c)), demonstrating the effect of the introduction of the stiffness coefficient. These experiments reveal a monotonic decrease in the maximum/average cartographic error and a monotonic increase in maximum/average polygon complexity based on the value of s_{high}.

Non-Triangulated Graphs. We tested our algorithm for graphs with ratio of removed edges $rem \in \{0, 0.2, 0.4, 0.6\}$ (Fig. 10 and 11). For each value of rem, we conducted experiments on fifty graphs, each with forty nodes, a nesting ratio of zero, and a weight ratio of five. We let

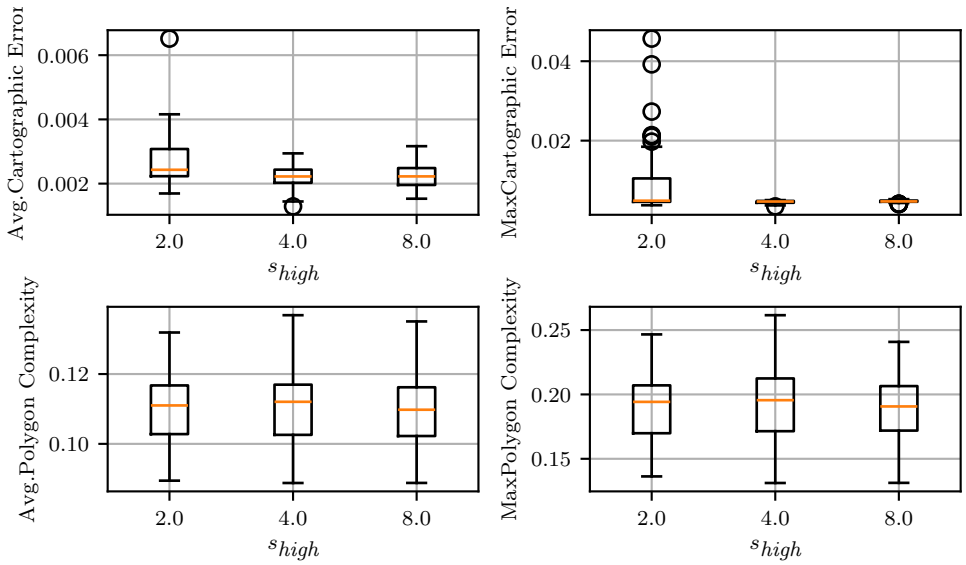
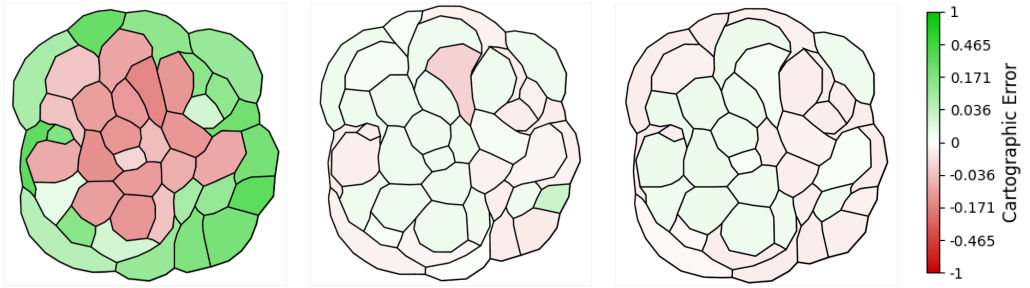


Figure 8 Cartographic Error and Polygon Complexity for different values of maximum stiffness, $n = 20$, $w = 5$, $nest = 0$.



(a) `MS-algorithm` **(b)** $s_{high} = 2$ **(c)** $s_{high} = 8$

Figure 9 Metaphorical maps for the same graph for `MS-algorithm` and `NEW-algorithm` for different values of max stiffness s_{high}.

the algorithm run for 1200 iterations. Recall that hole region weights were assigned according to Equation 3. Normalized cartographic error and polygon complexity were measured only for non-hole regions.

We observed that increasing the parameter *rem* (i.e., the fraction of removed edges) led to a slight increase across all four evaluation metrics (Fig. 10). This can be explained by the fact that the increasing number of missing edges creates more complex patterns of adjacency, i.e. holes behave as nodes of high degree.

We also observed that holes can have narrow passages; refer to Fig. 11.(c)-(d), which possibly can also be explained by the relatively high number of regions adjacent to a hole. We anticipate that this effect can be mitigated by using a different hole weight function and by re-engineering the applied forces.

The effect of the initial drawing. Given that we managed to, effectively, get the average cartographic error down to zero, we did not study the effect of the initial metaphorical map used by our algorithm on the cartographic error. Instead, we focused on the polygon

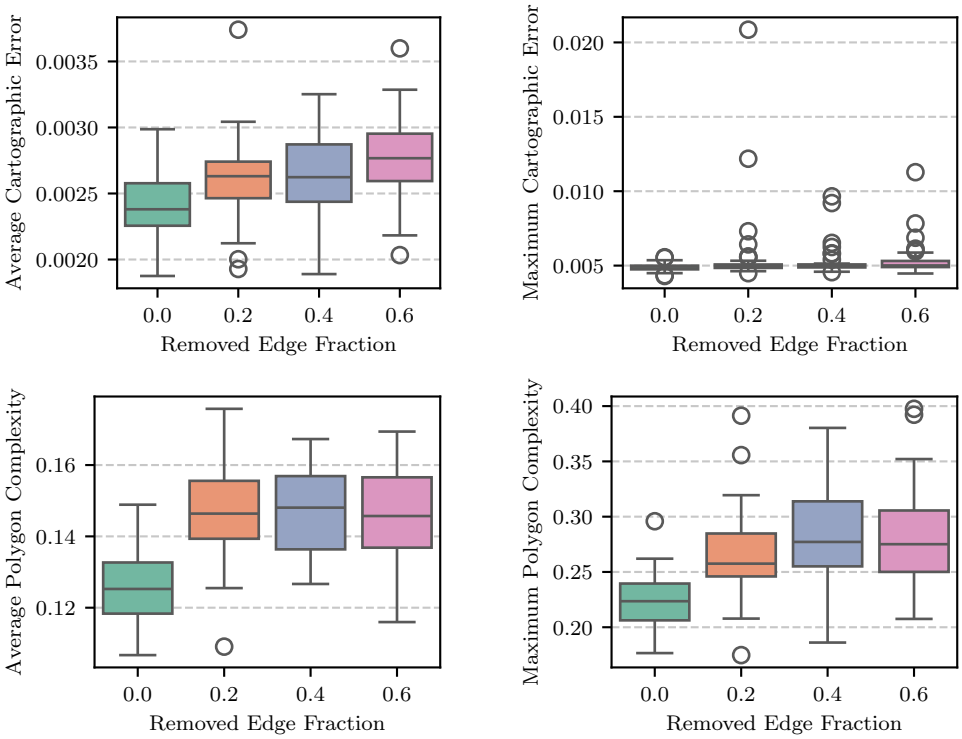


Figure 10 Cartographic Error and Polygon Complexity for different values of the ratio *rem* of removed edges and by adopting function *holeWeight*() for assigning weights to the polygonal holes. Here, $n = 40$, $w = 5$, $nest = 0$.

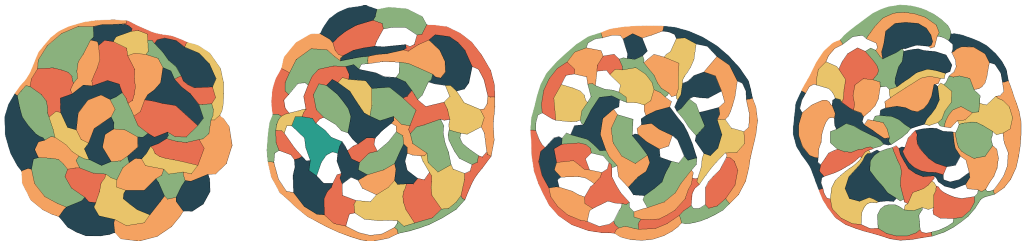
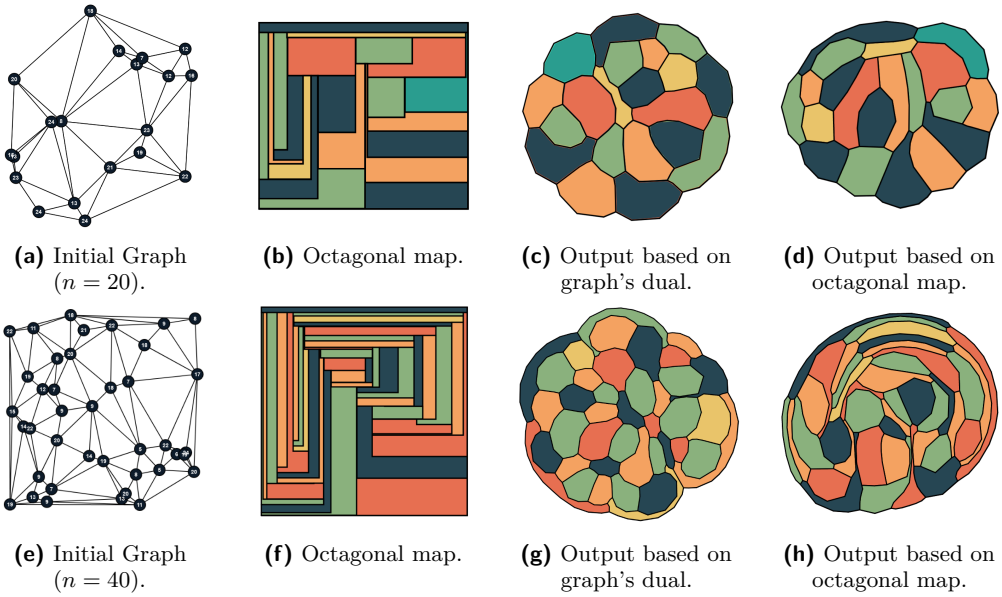


Figure 11 Example for the values of $rem \in \{0, 0.2, 0.4, 0.6\}$.

complexity and we examined the effect of having an initial map with good average cartographic error. As such good initial maps we considered the maps produced by the algorithm in [3] (referred to as *octagonal map*) which produces rectangular maps with at most 8 corners and of average cartographic error close to zero. We observed that, when using octagonal maps as initial layout, no consistent improvement was observed while the produced maps appeared to have long and skinny regions. Moreover, when the number of vertices becomes larger the metaphorical maps become harder to read due to their long, skinny, and circular shaped regions. See Fig. 12.(a-d) and Fig. 12.(e-h) for sample metaphorical maps of 20 and 40 regions, respectively. Finally, it is noted that our algorithm also required a larger number of iterations (about double) when initialized with such maps in order to converge to its final metaphorical map solution.



(a) Initial Graph ($n = 20$). **(b)** Octagonal map. **(c)** Output based on graph's dual. **(d)** Output based on octagonal map.

(e) Initial Graph ($n = 40$). **(f)** Octagonal map. **(g)** Output based on graph's dual. **(h)** Output based on octagonal map.

Figure 12 Metaphorical maps produced by our algorithm when the initial map is based on the graph's dual transformation and on an octagonal map. (a)-(d): 20 vertex graph. (e)-(h): 40 vertex graph.

5 Conclusion

In this work we have proposed an improvement of `MS-algorithm` [17] that produces metaphorical maps of vertex-weighted graphs and presented a detailed experimental evaluation of the newly proposed algorithm. Our goal was to close the gap between the metaphorical maps with perfect cartographic accuracy [2, 16, 24] that despite using a few corners per polygonal region can look quite complex and the simple-looking metaphorical maps produced by `MS-algorithm` with cartographic error up to 30%. Our experiments showed that we can achieve cartographic error close to zero, by paying a small price for the polygon complexity. Our algorithm achieves this improvement by using the notion of region stiffness that adapts as the force-directed simulation runs. We have seen that the maximum value of this stiffness determines the interplay between the cartographic error and polygon complexity.

To quantify the complexity of the maps, we applied the quality metric introduced in [8], who presented examples of geographic maps showing that this metric corresponds to the human intuition of map complexity. Our experiments with metaphorical maps do not contradict this intuition. However, we believe that whether this function is indeed appropriate for qualifying the human intuition of the metaphorical maps complexity has to be investigated in a user study.

We conjecture that the performance of the algorithm for non-triangulated graphs can be further improved by considering different hole weight functions and by engineering the forces to avoid narrow passages. Finally, it is also interesting to compare the quality of the maps for non-triangulated graphs with and without holes. We defer these investigations to the extended version of this work.

References

1 Muhammad Jawaherul Alam. *Contact representations of graphs in 2D and 3D*. Doctoral thesis, The University of Arizona, 2015. URL: https://www.proquest.com/openview/5bf2b2dc7f21bf9f8759bdc89632b50b/1?pq-origsite=gscholar&cbl=18750.

2 Muhammad Jawaherul Alam, Therese Biedl, Stefan Felsner, Andreas Gerasch, Michael Kaufmann, and Stephen G. Kobourov. Linear-time algorithms for hole-free rectilinear proportional contact graph representations. *Algorithmica*, 67(1):3–22, 2013. doi:10.1007/S00453-013-9764-5.

3 Muhammad Jawaherul Alam, Therese Biedl, Stefan Felsner, Michael Kaufmann, Stephen G. Kobourov, and Torsten Ueckerdt. Computing cartograms with optimal complexity. *Discrete & Computational Geometry*, 50(3):784–810, 2013. doi:10.1007/s00454-013-9521-1.

4 Muhammad Jawaherul Alam, Stephen G. Kobourov, and Sankar Veeramoni. Quantitative measures for cartogram generation techniques. *Comput. Graph. Forum*, 34(3):351–360, 2015. doi:10.1111/cgf.12647.

5 Marshall Bern and David Eppstein. *Mesh Generation and Optimal Triangulation*, pages 47–123. World Scientific Publishing Co Pte Ltd, 1995. doi:10.1142/9789812831699_0003.

6 Robert P. Biuk-Aghai, Patrick Cheong-Iao Pang, and Bin Pang. Map-like visualisations vs. treemaps: an experimental comparison. In Robert P. Biuk-Aghai, Jie Li, and Shigeo Takahashi, editors, *Proceedings of the 10th International Symposium on Visual Information Communication and Interaction, VINCI 2017, Bangkok, Thailand, August 14-16, 2017*, pages 113–120. ACM, 2017. doi:10.1145/3105971.3105976.

7 I. Bortins, S. Demers, and K. Clarke. Cartogram types, 2002.

8 Thomas Brinkhoff, Hans-Peter Kriegel, Ralf Schneider, and A. Braun. Measuring the complexity of polygonal objects. In Patrick Bergougnoux, Kia Makki, and Niki Pissinou, editors, *Proceedings of the 3rd ACM International Workshop on Advances in Geographic Information Systems, Baltimore, Maryland, USA, December 1-2, 1995, in conjunction with CIKM 1995*, page 109. ACM, 1995.

9 Rafael G. Cano, Kevin Buchin, Thom Castermans, Astrid Pieterse, Willem Sonke, and Bettina Speckmann. Mosaic drawings and cartograms. *Comput. Graph. Forum*, 34(3):361–370, 2015. doi:10.1111/cgf.12648.

10 Prabu David. Seeing is believing: Comparative performance of the pie and the bar. *Newspaper Research Journal*, 17(1-2):89–104, 1996. doi:10.1177/073953299601700109.

11 Mark de Berg, Elena Mumford, and Bettina Speckmann. On rectilinear duals for vertex-weighted plane graphs. *Discret. Math.*, 309(7):1794–1812, 2009. doi:10.1016/j.disc.2007.12.087.

12 Danny Dorling. Area cartograms: their use and creation. *University of East Anglia: Environmental Publications*, 59, 1996.

13 Marius Hogräfer, Magnus Heitzler, and Hans-Jörg Schulz. The state of the art in map-like visualization. *Comput. Graph. Forum*, 39(3):647–674, 2020. doi:10.1111/cgf.14031.

14 Yifan Hu. Efficient and high quality force-directed graph drawing. *Mathematica Journal*, 10:37–71, January 2005.

15 Eleni Katsanou, Tamara Mchedlidze, Antonios Symvonis, and Thanos Tolias. An algorithm for accurate and simple-looking metaphorical maps. Tech. Report arXiv:2508.19810, Cornell University, 2025. doi:10.48550/arXiv.2508.19810.

16 Linda Kleist. Drawing planar graphs with prescribed face areas. *J. Comput. Geom.*, 9(1):290–311, 2018. doi:10.20382/jocg.v9i1a9.

17 Tamara Mchedlidze and Christian Schnorr. Metaphoric Maps for Dynamic Vertex-weighted Graphs. In Marco Agus, Wolfgang Aigner, and Thomas Hoellt, editors, *EuroVis 2022 - Short Papers*. The Eurographics Association, 2022. doi:10.2312/evs.20221090.

18 Sabrina Nusrat and Stephen G. Kobourov. The state of the art in cartograms. *Comput. Graph. Forum*, 35(3):619–642, 2016. doi:10.1111/cgf.12932.

19 Steven J. Owen. A survey of unstructured mesh generation technology. In *7th International Meshing Roundtable*, 1998.

20 Bahador Saket, Carlos Scheidegger, and Stephen G. Kobourov. Comparing node-link and node-link-group visualizations from an enjoyment perspective. *Comput. Graph. Forum*, 35(3):41–50, 2016. doi:10.1111/cgf.12880.

21 C. Schnorr. Visualizing dynamic clustered data using areaproportional maps. Master's thesis, Karlsruhe Institute of Technology, 2020. URL: https://github.com/jenox/Master-Thesis.

22 Paolo Simonetto, Daniel Archambault, David Auber, and Romain Bourqui. Impred: An improved force-directed algorithm that prevents nodes from crossing edges. *Comput. Graph. Forum*, 30(3):1071–1080, 2011. doi:10.1111/j.1467-8659.2011.01956.x.

23 Roberto Tamassia, editor. *Handbook on Graph Drawing and Visualization*. Chapman and Hall/CRC, 2013.

24 Carsten Thomassen. Plane cubic graphs with prescribed face areas. *Comb. Probab. Comput.*, 1:371–381, 1992. doi:10.1017/S0963548300000407.

25 Waldo Tobler. Thirty five years of computer cartograms. *An. of the Assoc. of American Geographers*, 94(1):58–73, 2004. doi:10.1111/j.1467-8306.2004.09401004.x.

26 William Thomas Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 3(1):743–767, 1963.

27 Han-Ming Wu, ShengLi Tzeng, and Chun-houh Chen. *Matrix Visualization*, pages 681–708. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. doi:10.1007/978-3-540-33037-0_26.

28 yWorks GmbH. yfiles for html. https://www.yworks.com/products/yfiles-for-html, 2025. Software library.

Graph Drawing Contest Report

Sara Di Bartolomeo
TU Wien, Austria

Fabian Klute
Universitat Politècnica de Catalunya, Spain

Debajyoti Mondal
University of Saskatchewan, Saskatoon, Canada

Jules Wulms
TU Eindhoven, The Netherlands

Abstract

This report describes the 32nd Annual Graph Drawing Contest, held in conjunction with the 33rd International Symposium on Graph Drawing and Network Visualization (GD'25) at Linköping University, Norrköping, Sweden. The mission of the Graph Drawing Contest is to monitor and challenge the current state of the art in graph-drawing technology. This year's edition featured two categories, a creative topic in which participants visualized a dataset based on the Netflix show *Dark* and a live challenge held at the conference where participants had to draw a graph on a grid, such that the drawing is k-planar for as low a k as possible. A special feature of this year's contest is that the submissions to the creative topic were exhibited in the "Norrköping Decision Arena", a room with a circular annulus-shaped screen.

2012 ACM Subject Classification Human-centered computing → Visualization systems and tools; Human-centered computing → Visual analytics

Keywords and phrases Graph Drawing, Information Visualization, Graph Drawing Contest

Digital Object Identifier 10.4230/LIPIcs.GD.2025.41

Category Graph Drawing Contest Report

Acknowledgements The contest committee would like to thank the organizing and program committee of the conference; the organizers who provided us with a room with hardware for the live challenge and monetary prizes; Niko Reski who organized everything regarding the Decision Arena, including an interactive app for viewing and testing submissions; the generous sponsors of the symposium; and all the contestants for their participation. Further details including all submitted drawings and challenge graphs can be found on the contest website.

1 Introduction

This year, the Graph Drawing Contest was again divided into two parts, as was usual for the past years: the *creative topic* and the *live challenge*.

Following the last two editions, this year's creative contest focused on only one dataset. The creative topic was *Dark*: a German sci-fi thriller about time travel, fate, and family secrets. The dataset consists of events in the series, characters involved in each event, and some additional information per event (as described below). The contestant are invited to explore the intricate relationships between events, characters, and timelines in Dark. The data set was published about half a year in advance, and contestants submitted their visualizations before the conference started.

The live challenge took place during the conference in a format similar to a typical programming contest. Teams were presented with a collection of *challenge graphs* and had one hour to submit their highest scoring drawings. Since the topic had remained the same for


© Sara Di Bartolomeo, Fabian Klute, Debajyoti Mondal, and Jules Wulms;
licensed under Creative Commons License CC-BY 4.0



the last two years, we introduced a new optimization problem this year: given an undirected simple graph and a grid, find a straight-line drawing with vertices placed on grid points, such that the drawing is k-planar for as low a k as possible.

Overall, we received 34 submissions: 12 submissions for the creative topics and 22 submissions for the live challenge (14 manual and 8 automatic).

2 Creative Topic

The dataset for the Creative Topic represents the events of the Netflix show *Dark*. We provided a directed graph, in which each event forms a vertex and edges represent cause and effect relationships between events.

Each vertex and edge is enriched with some metadata to flesh out the events and the relationships between them. Like last year, we explicitly asked contestants to formulate a question or hypothesis about the data and try to answer or explore it with their visualization.

Vertices have an `ID`, a `Description`, and a `Date`. The date is especially important since time travel plays an important role in Dark. Furthermore, we provide a list of `Characters` that are involved in an event. We added some additional metadata about whether a character dies and/or whether an event is a pivotal moment in the story. Finally, as the show progresses, the characters find out there are multiple parallel worlds, so for each event we explicitly state which `World` it takes place in. This lead to the following features for each event:

```
{
    "ID": "Incremental ID for each event",
    "Description": "Textual description of the event",
    "Date": "Date of the event in (dd-mm-yyyy) format",
    "Important Trigger": "Boolean value indicating if the
       event is an important trigger for the series",
    "Death": "Boolean value indicating if the event involves a
        death",
    "Characters": "List of characters involved in the event",
    "World": "World in which the event takes place"
}
```

Edges have a slightly simpler format, with an `ID` and vertex/event IDs for `Source` and `Target` vertices of each directed edge between events. The additional metadata consists of an edge `Type`, which included things such as "Successful time travel" or "World swap". Finally, certain edges have a `Description` if some elaboration on the relation between events was necessary. The features of edges can then be described as:

```
{
    "ID": "Incremental ID for each edge",
    "Source": "ID of the source event",
    "Target": "ID of the target event",
    "Type": "Type of relationship between the events",
    "Description": "Few edges include a description of the
       relationship"
}
```

The total graph had 554 vertices and 628 edges.

The raw data forming the basis of the dataset was extracted from a Reddit post on `r/DarK` by user `u/aldersonloop59`.[1] The Reddit post refers to an interactive Lucidchart, a tool to create diagrams, such as flowcharts and mindmaps. While this Lucidchart document of Dark is very comprehensive, Lucidchart itself is just a visualization tool, and not meant to internally represent complex graph structures with metadata. As such, the XML output generated by Lucidchart could not capture all the relational information and metadata visualized in the document. Hence, the contest committee took on the labor-intensive task of manually extracting all the data into two tables for vertices and edges. In the process some wrong information was corrected and many bits and pieces were refined — using the committee's knowledge of the Dark plotlines — to create the final dataset.

We received 12 submissions for the creative challenge. Submissions were evaluated according to four criteria:

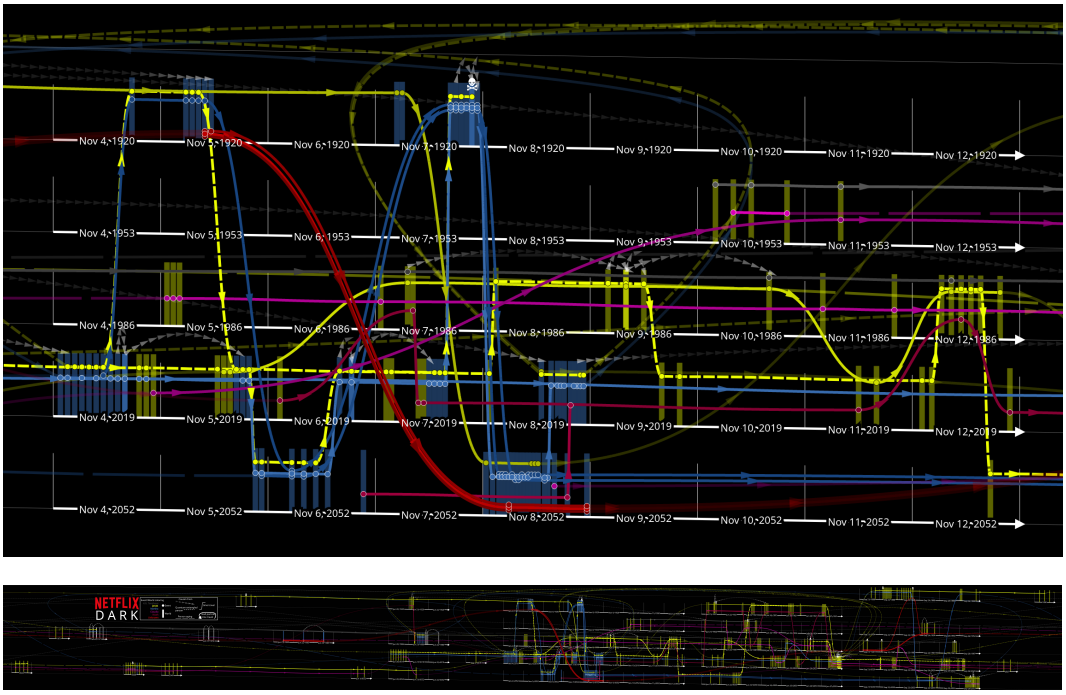
1. readability and clarity of the visualization,
2. aesthetic quality,
3. novelty of the visualization concept, and
4. design quality.

In practice, we asked ourselves the following questions to assess these criteria: **(1)** How much does this submission help you understand (parts of) the data? **(2)** Do the aesthetics fit the theme of Dark? **(3)** Does the submission require (involved) graph drawing techniques? And this year especially: how well is the circular screen utilized? **(4)** How coherent and polished is the submission? As is the case every year, we noticed that it is a difficult decision to choose one submission over the other, and it takes a lot of discussion to weigh all the criteria against each other and come to a definitive top 3. To ease this discussion and to keep the creative topic managable at the conference, we selected a top six out of all submissions before the conference. This top six was exhibited in the Norrköping Decision Arena during a dedicated session at the Graph Drawing Symposium. We also made all the submissions available on the contest website in the form of a virtual poster exhibition. For a complete list of submissions, refer to `https://www.graphdrawing.org/gdcontest/2025/results/`.

[1] `https://www.reddit.com/r/DarK/comments/hqnzgf/spoilers_s3_d_a_r_k_the_complete_visual_timeline/`

3rd Place: Tom Baumgartl, Luke Watkin, George Bell, Daniel Archambault, Alma Cantu and Tatiana von Landesberger – Visualizing Time Travelling, Worlds, and Quantum Variants: A Storyline-Spiral Layout of Netflix Dark

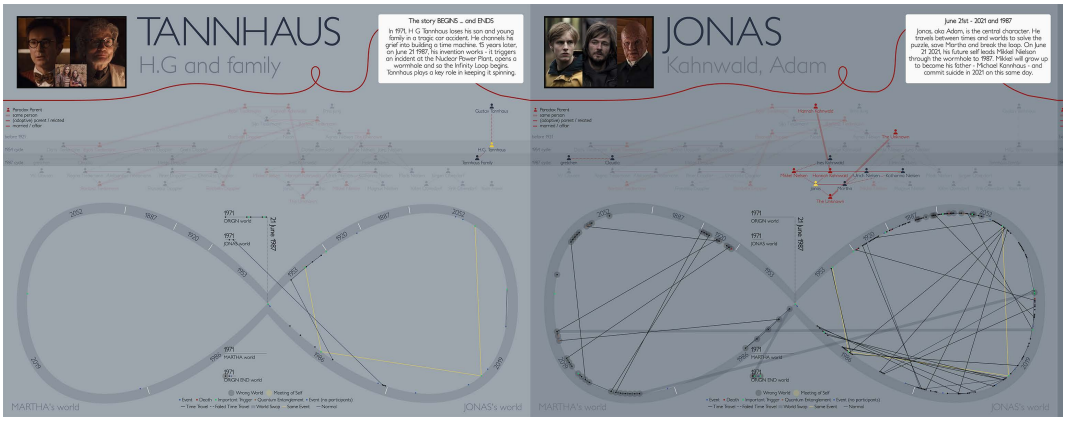
The team submitted a storyline visualization that made excellent use of the circular screen — a stand-out feature for this year's contest.

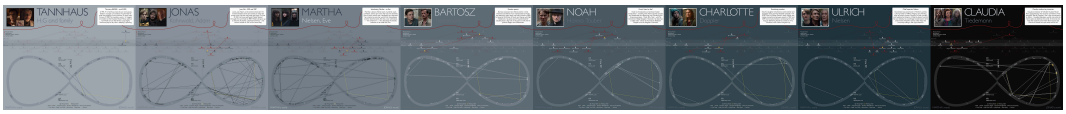


> Our visualization tackles the complex narrative of Dark, where characters exist across parallel worlds, multiple ages, and quantum variants connected by time travel. We designed a static storyline spiral layout, splitting time into 33-year increments stacked on a sloped spiral to emphasize the cyclic theme. Character variants are shown as identity-colored continuous lines with fixed vertical ordering to reduce crossings. Scenes from different worlds appear as colored vertical bars, with dots marking participation and saturation encoding event importance. Time travel arcs, quantum entanglements, and uncertainty between events are encoded through curves, dashed lines, and opacity, revealing both local detail and the global narrative structure.

2nd Place: Bryony Miles – Dark Contest – Paradox Tree and Infinity Loop

The submission by Bryony had outstanding aesthetic and design quality; the use of the infinity sign was fitting, since it plays a central role in Dark, and the recurring design looked stunning on the circular screen.

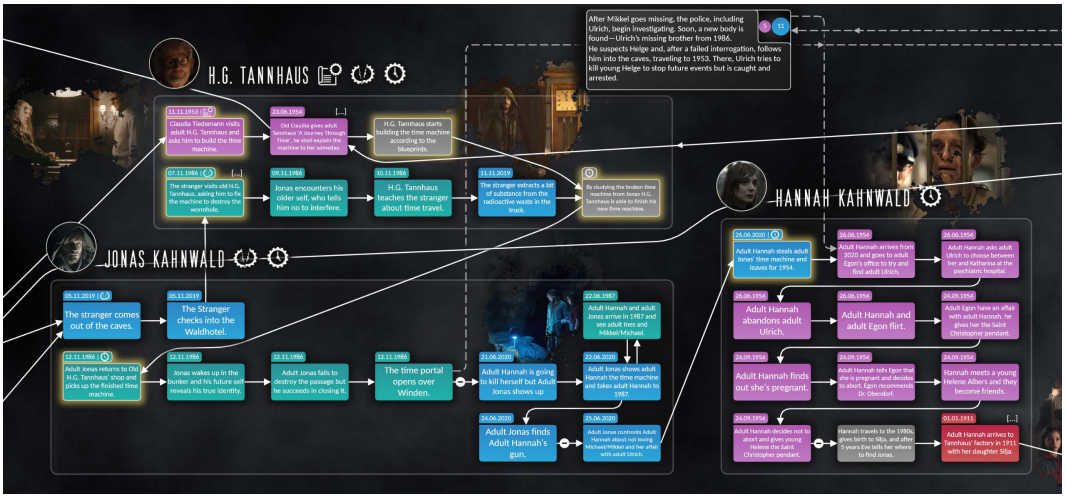




> My final design aimed to fulfil four goals:
>
> 1. Show the main story arc across the four worlds (ORIGIN world -> stuck in the Infinity Loop (JONAS + MARTHA world) -> ORIGIN END world)
> 2. Tell the story through 8 key characters and their nodes + edges
> 3. Use the unique 360 degree viewing platform to connect the characters and compare interactions
> 4. Build a family tree to visually illustrate the many Parent Paradoxes in the plot
>
> I also wanted to use a colour palette and imagery drawn from the Netflix series - character photos, Infinity Loop, red thread connecting the characters. D3.js was always going to be my tool of choice. The trickiest technical challenges were running the nodes along the SVG path and syncing the animations on the interactive version.

1st Place: Florian Saß, Jakob Speitkamp and Guilherme Monteiro Oliveira – Journey of a Time Machine.

This team gave a unique twist to their exposition of the data, choosing to follow the time machine throughout the story. This idea was well-executed and resulted in an exciting and polished submission.





The comments of the authors of this submission are included as a separate report.

3 Live Challenge

As always, the live challenge took place during the conference and lasted exactly one hour. During this hour, participants of the conference could take part in the manual category (in which they could attempt to draw the graphs using a supplied tool: `http://graphdrawing.org/gdcontest/tool/`), or in the automatic category (in which they could use their own software to draw the graphs).

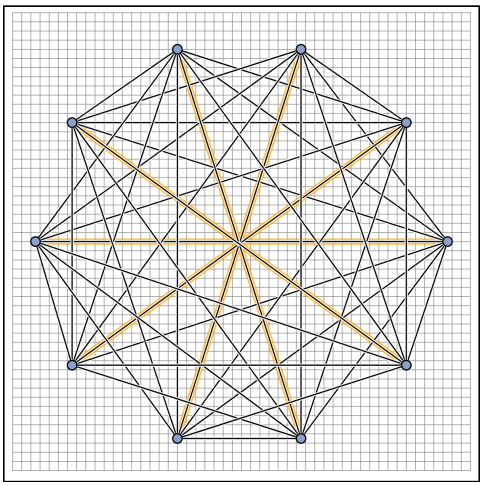
We introduced a new rule this year, allowing teams to participate only if they have a representative attending the conference. This representative can be a team member, or another (closely related) member of the GD community, such as a supervisor (of student team), collaborator, or friend. The contest is organized for the GD community, and hence we think it is important that the participants interact with each other and with the community, to share knowledge about how to efficiently solve the challenges in the contest.

To coordinate the contest, we gave a brief introduction, answered questions, and gave participants the possibility to form teams. We were kindly provided with both a room in the conference building, and laptop facilities, to accommodate to the needs of the participating teams. The largest automatic contest instance gave some teams technical issues, as the server would sometimes have problems storing the solutions sent in by teams. Luckily resending the solutions would resolve this issue, and the contest was otherwise uneventful in terms of technical issues.

The challenge of this year's contest was to draw a given undirected graph on a given grid, while ensuring that the straight-line drawing was k-planar for as small a k as possible. We did not allow for points to lie on (the interior of) edges. For each edge we count the number of crossings, and the edge with the largest number of crossings for a single edge determines the quality metric k of a drawing. The study of k-planarity and k-planar graphs is an active field of research within graph drawing, and hence this contest topic forms a timely challenge to the community to put the obtained knowledge into practice.

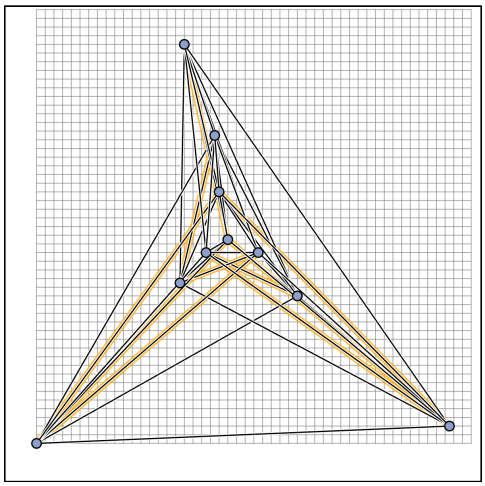
3.1 The Graphs

In the manual category, participants were presented with seven graphs. While the graphs are usually arranged (roughly) from small to large, this year there are quite some smaller graphs in the later half of the manual category. Instead the graphs are ordered such that it is increasingly less obvious what the underlying structure is of the graph, such that it becomes harder to minimize k. The automatic category has an entirely different set of nice instances, as opposed to last years, where they solve the manual category instances and in addition another eight larger graphs. To counteract the decrease in number of instances, the size of the final few instances has increased significantly. The graphs were constructed to have different structures, and for the automatic category, they are roughly ordered from small to large.

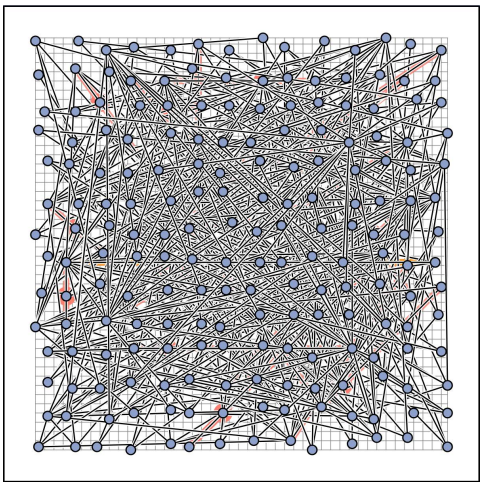


Provided drawing in manual category

16-planar

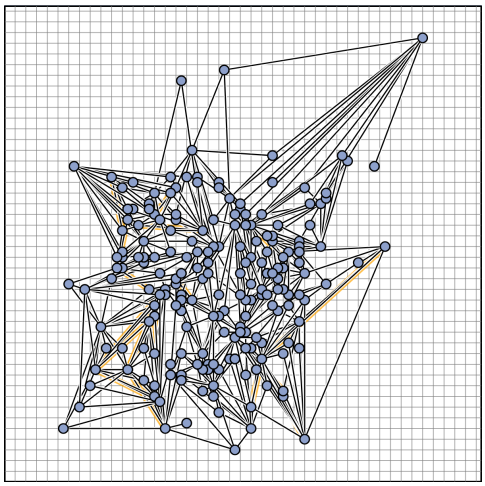


Best manual solution
The Untanglers
5-planar



Provided drawing in automatic category

Invalid drawing



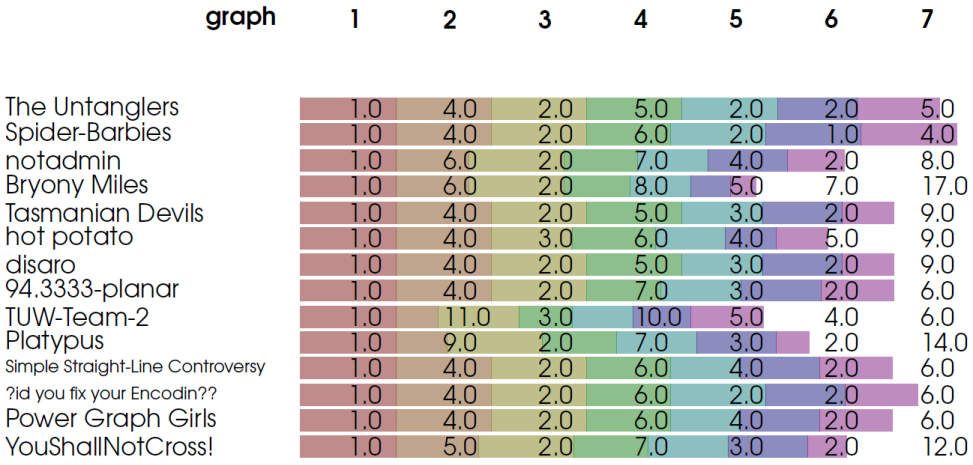
Best automatic solution
OMEGA
6-planar

For illustration, we include one graph of each category, showing both the initial drawing as well as the best solution found by one of the participating teams. These instances show both strengths and weaknesses of optimizing k-planarity. For the manual contest instance, the initial drawing shows the symmetry of the graph, and all crossings are easily recognized because the drawing has few small and/or thin faces. The drawing with lower k-planarity loses all these features. However, for the automatic category, the initial drawing a a big hairball, and the best solution clearly shows more structure in the graph. While in previous years, the committee saw repeatedly that manual (human) drawings of graphs display a deeper understanding of the underlying graph structure than automatic and therefore have better readability. For this year's contest that is not always the case, as becomes clear from the examples above. Still, the committee is curious to see how this evolves in the next year, as the live contest topic remains the same for a second year, as is common for the contest.

For the complete set of graphs and submissions, refer to the contest website at `https://www.graphdrawing.org/gdcontest/2025/results/`. The graphs are still available for exploration and solving on the Graph Drawing Contest Submission System: `https://www.graphdrawing.org/gdcontest/tool/`.

3.2 Results: Manual Category

Below we present the full list of scores for all teams. The numbers listed are the k values for the lowest k-planarity achieved by the drawings; the horizontal bars visualize the corresponding scores, where lower scores correspond to larger bars.



Third place: **Did you fix your Encoding?**, consisting of Soeren Terziadis and Birgit Vogtenhuber.

> Creating an entirely new embedding from scratch appeared to be hard. Instead we mostly focussed on iteratively resolving crossings on the crossing-heaviest edges. Long edges all pointing in roughly the same direction seemed suboptimal but the solution of placing them more central tended to crowd the center. So we focussed on creating sufficient space on the inside to realize small moves. Whenever a heavy edge was part of a long subdivided path of degree 2 nodes we often did not change the actual routing of the path but instead equally distributed the crossings along the path.
> *Soeren Terziadis and Birgit Vogtenhuber*

Second place: **The Untanglers**, Joseph Dorfer, Ly Orgo, and Benedikt Hahn.

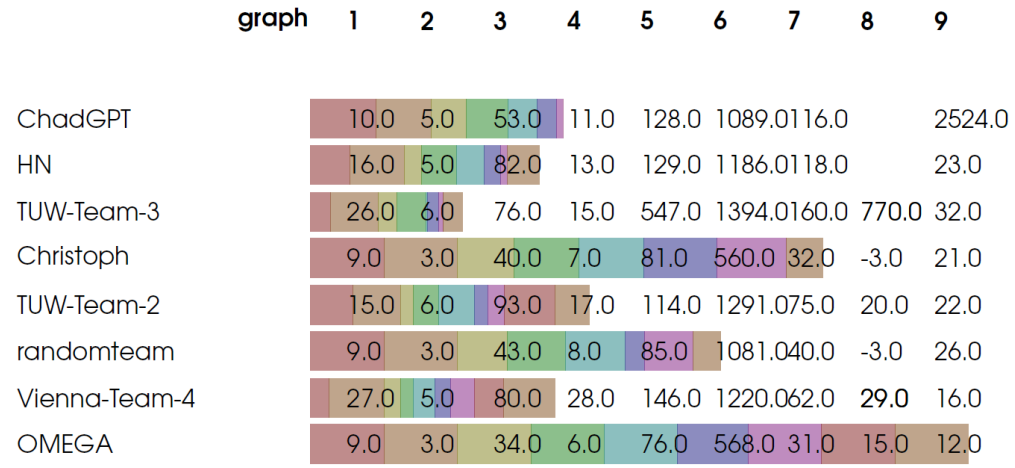
> To untangle the instances, we started by getting a grasp on the structure of the graph. For example, one of the instances was a near-complete graph, so we tried to mimic a crossing-optimal straight-line drawing of the complete graph (i.e., tripod-like). Then, the person in charge of the mouse could focus on minor local improvements, while the other two suggested more substantial local changes, involving redrawing multiple vertices. At the same time, we tried to redraw areas that seemed too crowded. Unfortunately, just as the timer dropped to zero, we missed a further improvement by a few seconds.
> *Joseph Dorfer, Ly Orgo, and Benedikt Hahn*

Winner: **Spider-Barbies**, consisting of Giordano Andreolna, Philipp Kindermann, and Allegra Strippoli.

> Before the contest even began, the toughest challenge was finding a team name. In the end, we combined our favorite superheroes; and thus, the Spider-Barbies were born. Inspired by the beautiful waterfall of Norrköping, our team of two rookies and one man freshly out of retirement faced seven graphs, each edge demanding fewer crossings than a catwalk runway. We modeled the problem through spider webs, aiming to maximize the mosquitos caught, a well-known natural reduction. Fueled by coffee, quick sketches, and a bit of spider silk magic, we tangled our way through layouts. When the clock struck, our graphs stood polished, strutting like Barbies on stage – and so did we.
> *T. Hegemann and J. Zink*

3.3 Results: Automatic Category

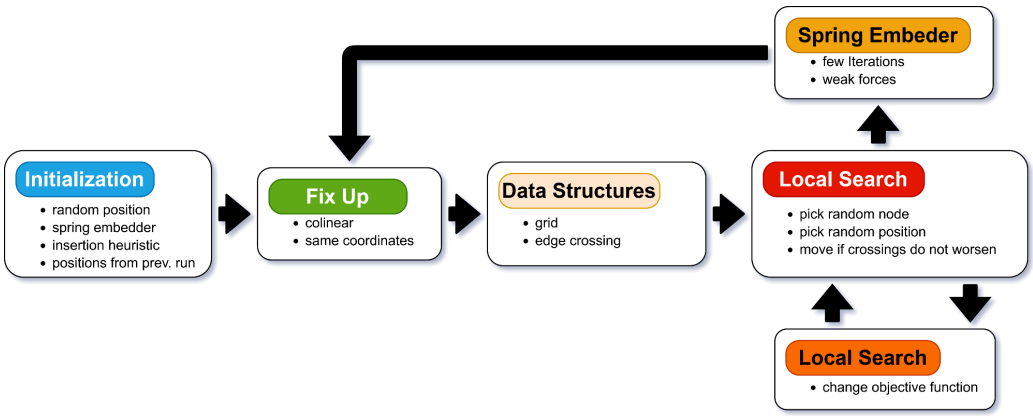
In the following we present the full list of scores for all teams that participated in the automatic category. The numbers listed are again the k values for the lowest k-planarity achieved by the drawings; the horizontal bars visualize the corresponding scores (lower scores lead to longer bars).



Third place: **randomteam**, consisting of Stefan Brandmair and Luca Marius Cobzaru.

> For this contest, we employed a simple pipeline of initialization, followed by local optimisation. It turns out that spring embedders are king. For local optimisation, we chose the simplest possible "move node" heuristics. To make it fast, we coupled it with a few heuristics for early exiting, such as the order in which we count edges after moving. For the graph, we made sure to write our own specialized array-based type, instead of using a general Julia library. After implementing an incremental crossing computation, we also added a grid for quickly finding nearby nodes. Finally, we created an open testing tool to compare our implementations.
> *Stefan Brandmair and Luca Marius Cobzaru*

Randomteam also provided a schema to explain their process:



Second place: **Christoph**, consisting of Christoph Weber.

> My most effective strategy was to first generate a solid starting point using multiple force-based algorithm runs. From there, I iteratively improved the layout with a local search approach. I found that incorporating randomization for node selection and positioning during this search generally produced the best results. The main challenge was making my program efficient enough to test a sufficient number of positions in a reasonable time. I achieved this with a grid indexing system and cross-computation strategies that abandoned unpromising moves early. My local search normally rejected any move that increased k-planarity. To escape local minima, the search would temporarily accept a controlled worsening of the solution before resuming.
> *Christoph Weber*

Winner: **OMEGA**, consisting of Julien Bianchetti and Laurent Moalic.

The report from team OMEGA is attached to this report as its own standalone report.

4 Conclusion

The 2025 edition of the Graph Drawing Contest was another success in both participation and result. While the attendance to the GD conference was a bit lower than in Vienna (2024), the total number of creative topic submissions and participating teams in the live challenge went up. The participation in the live challenge remained strong, even though the committee introduced a new rule that requires each team to have a representative at the conference. This is a good sign, as it shows that there are enough participants in the community, or the participants from outside the community are willing to attend to conference. As such, this new rule achieves exactly the intended outcome: it creates engagement and discourse within the Graph Drawing community and attracts outsiders to participate in our community. The opportunity to use the Decision Arena in Norrköping was also a highlight of his year's contest. Participants loved seeing the submissions in an entirely new and unusual format, and the committee is really grateful to the organizers for helping us to make this happen. The committee will look out for more such opportunities in the future.

Journey of a Time Machine

Florian Sass
TU Wien, Austria

Jakob Speitkamp
TU Wien, Austria

Guilherme Monteiro Oliveira
TU Wien, Austria

Abstract

What does the story of Netflix's Dark look like from the perspective of the suitcase time machine? We visualized this perspective with our contribution to the Graph Drawing Contest Creative Topic. We outline the used ideas, concepts and give a high-level overview over the algorithmic implementations.

2012 ACM Subject Classification Human-centered computing → Graph drawings

Keywords and phrases graph drawing, Dark, grid-based graph drawing, 360 degree visualization, cyclic graph drawing

Digital Object Identifier 10.4230/LIPIcs.GD.2025.42

Category Graph Drawing Contest Abstract

1 Main Concept

We are interested in the interactions with time machines in the story of Dark. Therefore, we visualize the journey of the suitcase time machine, a central method of time travel in the first two seasons of Dark. It traces the machine's path from character to character, showing how it is acquired, lost, created, and how it eventually breaks down. The graph highlights key, life- defining moments connected to the machine and emphasizes its cyclical nature, reflecting the show's core themes. Our concept is mostly inspired by and based on already existing material on the topic, like YouTube videos [5] [6] and wiki articles [4].

To bring this concept to life, we defined several core components for our graph:

1. **Blocks:** Each block represents a period when a specific character possesses the time machine. It's a group of nodes showing how they acquire it, their experiences, and how it's eventually lost or passed on.
2. **Time-period color coding:** Nodes within a block are color-coded based on the time-period in which the events occur. A change in color between nodes typically indicates time travel, though not necessarily via the suitcase time machine.
3. **Life stages:** We illustrate the time machine's three life stages: blueprint, working, and broken. These are shown at a high level for each character and are marked on specific nodes where a change occurs.
4. **Trigger events:** These are key moments in the narrative of the time machine itself. It's important to note that these are distinct from the dataset's predefined important triggers. Trigger events are shown by a golden glow around a node.

2 Implementation and Elements

We assigned story-relevant nodes to blocks and generated a separate grid layout for each. These blocks were manually arranged on a shared canvas, combining manual control of the overall layout with automatic placement of nodes within blocks. Edge routing was automated, minimizing collisions with other elements.



© Florian Sass, Jakob Speitkamp, and Guilherme Monteiro Oliveira;
licensed under Creative Commons License CC-BY 4.0



2.1 Libraries and Tools

We built the visualization as an SVG using D3 [1] and D3-Plus [2], with Node.js [3] for file handling and setup. This powered a simple web app that visualized the graph, allowed real-time adjustments to key variables via an UI, and supported interactive click-and- drag for certain elements. We also integrated the preview tool for one-click 360° viewing.

2.2 Data handling and Node selection

To select the relevant nodes for out story we first visually identified them in the already existing visualization and then traversing the dataset to mark all relevant nodes. Since some details were missing, we supplemented nodes, referred to as context nodes. In total, we ended up with 126 nodes (15 new) and 142 edges (35 new).

2.3 Node placement

To determine the position of each node within a block, we first performed a topological sort on all nodes in that block. We then used the resulting order, along with some logic, to assign positions within the block grid. We introduced a parameter to independently adjust the aspect ratio of each block to have more control over the layout.

2.4 Edge routing

To route the edges, we start with straight lines between nodes, which are passed to a shaping pipeline with several key steps:

- collision avoidance with blocks, nodes and summary nodes
- reduction of excess routing points, which were introduced by collision avoidance
- evenly spacing edges connected to a single node side
- smoothly bending the edge line at routing points

Because the contest format was a 360° display, an edge will use the shorter path towards a node. If an edge crosses the image border (left or right), the edge is split into two sub-edges, one for each side.

2.5 Summary Nodes

To provide more context, we added summary nodes that represent related, non-selected nodes within the broader storyline. Each includes a bubble chart showing the number of summarized nodes per time-period. Their edges follow a separate routing along the top to maintain focus on the main storyline.

References

1 D3. Accessed on October 13, 2025. URL: https://d3js.org/.
2 D3plus. Accessed on October 13, 2025. URL: https://github.com/d3plus/d3plus.
3 Nodejs. Accessed on October 13, 2025. URL: https://nodejs.org/en.
4 Tannhaus's clockwork device, 2023. Accessed on October 13, 2025. URL: https://dark-netflix.fandom.com/wiki/Tannhaus%27s_clockwork_device.
5 Dark & 1899. Dark | portable time machine timeline explained, 2020. Accessed on October 13, 2025. URL: https://www.youtube.com/watch?v=xnSta5jbN6M.
6 OneTake. Dark time machine timeline explained | dark netflix season 3, 2020. Accessed on October 13, 2025. URL: https://www.youtube.com/watch?v=hRM2KjU_OjQ.

Winning the GD Challenge for the 4th Time: Our Approach

Julien Bianchetti
Université de Haute-Alsace, IRIMAS UR 7499, F-68100 Mulhouse, France

Laurent Moalic
Université de Haute-Alsace, IRIMAS UR 7499, F-68100 Mulhouse, France

Abstract

We present the approach we designed to tackle and win the 2025 Graph Drawing Challenge on minimizing the k-planarity of graphs. Our method employs a multi-stage heuristic centered around two Simulated Annealing (SA) algorithms: the first aims to reduce the total number of crossings, while the second improves the k-value. To obtain a good initial solution, we first applied tools from the OGDF library, which helped reduce crossings. The challenge consisted of nine instances to optimize. Our approach achieved the best results on eight out of nine instances - sharing the top score twice with other teams and ranking first alone in six cases.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Human-centered computing → Graph drawings; Mathematics of computing → Combinatorial optimization

Keywords and phrases Graph Drawing Contest, simulated annealing, k-planarity

Digital Object Identifier 10.4230/LIPIcs.GD.2025.43

Category Graph Drawing Contest Abstract

1 Introduction

The 2025 Graph Drawing Contest live challenge addresses the problem of minimizing the parameter k in k-planar straight-line drawings of graphs. Given an input graph, the task is to compute a drawing in the plane where each edge is crossed by at most k other edges, and the objective is to minimize this maximum crossing number.

The algorithm we developed, named SAkGD, is based on two distinct Simulated Annealing (SA) phases, each with a specific fitness function. SA is a powerful metaheuristic for combinatorial optimization [7, 1] and has been successfully applied to aesthetic graph drawing in the past [3]. A key feature of our approach is that while each SA phase has its own parameters, they are not instance-dependent; we applied the same algorithm and parameters to every instance of the challenge.

2 General presentation of SAkGD: the SA based algorithm for k-planarity

The proposed algorithm is based on three main steps summarized as fallows:

1. Generation of a «not so bad» starting solution with OGDF
2. Improve the starting solution by removing as many crossing edges as possible
3. Optimize the improved solution in terms of k-value

2.1 Step 1: Generating a Not-So-Bad Starting Solution

For this step, our goal was to quickly generate a high-quality initial layout. We utilized the powerful Open Graph Drawing Framework (OGDF) [2], which provides robust implementations of several layout algorithms. We specifically used Stress Minimization [4] and the Fast Multipole Multilevel Method (FMMM) [6, 5].


© Julien Bianchetti and Laurent Moalic;
licensed under Creative Commons License CC-BY 4.0



Stress Minimization is applied only once, since it is deterministic – that is, two runs produce exactly the same result. Then, we apply FMMM as many times as possible within the first minute. Because FMMM is stochastic, two runs yield different solutions.

At the end of the first minute, or after the time required for Stress Minimization (SM) and FMMM, we consider the best solution obtained so far among the initial solution (directly from the data file), the one from SM, and all the FMMM runs. Here, the best solution is defined as the one with the smallest k-value.

2.2 Step 2: A 10-Minute SA for Crossing Reduction

The 2023 and 2024 contest topics focused on minimizing the total number of edge crossings in a graph. Another difference was that nodes could only be placed at predefined positions (locations). It is quite clear that reducing the number of crossing edges is relevant in terms of k-planarity. Indeed, even though decreasing the number of crossings is not necessarily synonymous with decreasing the k-value, it generally provides significant improvement in most cases.

Algorithm 1 describes the key steps for reducing the total number of edge crossings.

Algorithm 1 SA-based approach for removing edge crossings

Input: $initTemp$ the starting temperature, $decreaseTemp$ the decreasing temperature after each step, $decreaseTempWave$ the decreasing temperature after each wave, $tempLimit$ temperature limit before stopping, s_0 the initial solution
Output: $s*$ the best solution found so far

$s* \leftarrow s_0$
$k \leftarrow$ the starting k value from s_0
$crossing \leftarrow$ the total number of crossings from s_0
$kBest \leftarrow k$, $crossingBest \leftarrow crossing$
$startingTemp \leftarrow initTemp$
while $k > 0$ and $startingTemp > tempLimit$ **do**
  $currentTemp \leftarrow startingTemp$
  while $k > 0$ and $currentTemp > tempLimit$ **do**
    $node \leftarrow$ selectNode()
    $newPlace \leftarrow$ selectPlace()
    if acceptMove($node$, $newPlace$)==true **then**
      move($node$, $newPlace$) {update $crossing$ and k}
      if $k < kbest$ or $k == kbest$ and $crossings < crossingsBest$ **then**
        $kBest \leftarrow k$, $crossingBest \leftarrow crossing$, $s* \leftarrow currentSol$
      end if
    end if
  end while
  $startingTemp \leftarrow startingTemp * decreaseTemp$
  $currentSol \leftarrow s*$
end while
return $s*$

selectNode() and *selectPlace*() play a significant role. *selectNode*() is important because not all nodes have the same impact on the current k-value. If all edges originating from a given node have few crossings, moving that node will yield only a small improvement compared to moving a node connected to edges with many crossings. For this reason, nodes belonging to edges with many crossings are given a higher probability of being selected for movement.

selectPlace() is also important. If the new position selected for a node is far from its previous location, there is a risk of creating many new crossings. For this reason, we decided to give a higher probability of selection to positions close to the node's previous location.

A move is always accepted from *acceptMove*() if it reduces or maintains the total number of crossings. A move that increases the crossings by ΔC is accepted with a probability of $\exp(-\Delta C/T_{current})$.

In this phase, we focus on optimizing the total number of crossings. However, it should be noted that each new wave starts from the best k-solution found so far. This is done to prepare for the final and main step of the proposed algorithm.

2.3 Step 3: Remaining Time for k-Value Optimization

With the remaining time (at most 49 minutes), we switch to a second SA algorithm that directly optimizes the k-value. This phase uses the same structure as Algorithm 1 but employs a more sophisticated fitness function for the *acceptMove*() criterion:

1. Primary Objective: Local k-value Improvement. A move is evaluated based on the change in the maximum number of crossings on any edge involved in the move (i.e., edges incident to the moved node and edges that cross them). This local computation is much faster than re-evaluating the global k-value.
2. Secondary Objective: Total Crossing Number. If a move does not change the local maximum k-value, the decision is based on the change in the total number of crossings, as in the previous phase.

A move is accepted if it improves the solution according to this objective. Worsening moves are still accepted based on the same probabilistic formula, using the fitness change as ΔE. This dual-objective approach allows the search to fine-tune the placement of nodes that are bottlenecks for the k-value, while still encouraging a general reduction in crossings.

3 Application, parameters and results on the 2025 GA contest dataset

Table 1 presents the parameters used during the contest. These were not tuned per-instance, demonstrating the robustness of the approach. It could be interesting to tune them, especially based on instances characteristics.

Table 1 SA parameters for minimizing the total crossings number and the k-value

	initTemp	*decreaseTemp*	*decreaseTempWave*	*tempLimit*
min cross	50.0	0.999	0.99	0.01
min k	1.0	0.9999	0.99	0.01

Our method performed exceptionally well, securing the top score on 8 of the 9 contest instances. Table 2 summarizes these results.

Table 2 Performance on the 2025 Graph Drawing Contest dataset.

Instance	Our k-value	Best k-value	2nd Best k-value	3rd Best k-value	Our Rank
graph 1	**9**	9	9	9	1st (Shared)
graph 2	**3**	3	3	3	1st (Shared)
graph 3	**34**	34	40	43	1st
graph 4	**6**	6	7	8	1st
graph 5	**76**	76	81	85	1st
graph 6	**568**	560	568	1081	2nd
graph 7	**31**	31	32	40	1st
graph 8	**15**	15	20	29	1st
graph 9	**12**	12	16	21	1st

4 Conclusion

We presented SAkGD, the multi-stage heuristic that won the 2025 Graph Drawing Challenge. Its success stems from a three-step process: a fast generation of a good initial layout using OGDF, followed by two specialized Simulated Annealing phases. The first phase effectively reduces the total crossing number, creating a promising search space, while the second phase aggressively targets the primary objective of minimizing the k-value using a fitness function. The use of instance-independent parameters highlights the general applicability of our method. Future work could explore adaptive parameter tuning based on graph properties or hybridizing the SA with other local search techniques.

References

1 Dimitris Bertsimas and John Tsitsiklis. Simulated Annealing. *Statistical Science*, 8(1):10–15, 1993. doi:10.1214/ss/1177011077.

2 Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W. Klau, Karsten Klein, and Petra Mutzel. The open graph drawing framework (ogdf). *Chapter 17 in: R. Tamassia (ed.), Handbook of Graph Drawing and Visualization, CRC Press.*, 2014.

3 Ron Davidson and David Harel. Drawing graphs nicely using simulated annealing. *ACM Trans. Graph.*, 15(4):301–331, October 1996. doi:10.1145/234535.234538.

4 Emden Gansner, Yehuda Koren, and Stephen North. Graph drawing by stress majorization. In *Graph Drawing*, volume 3383, pages 239–250, September 2004. doi:10.1007/978-3-540-31843-9_25.

5 Martin Gronemann. Engineering the Fast-Multipole-Multilevel method for Multicore and SIMD architectures. Master's thesis, Technische Universität Dortmund, Dortmund, Germany, 2009.

6 Stefan Hachul and Michael Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. In *Graph Drawing*, volume 3383, pages 294–305, September 2004. doi:10.1007/978-3-540-31843-9_29.

7 S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983. doi:10.1126/science.220.4598.671.

Recovering Graphs from Their Witness Unit Square Representation

Maarten Löffler
Department of Information and Computing Sciences, Utrecht University, The Netherlands

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

Soeren Terziadis
Algorithms Group, TU Eindhoven, The Netherlands

Abstract

A wUSR of a graph G is a set of unit squares in the plane, one per vertex, if two vertices have an edge in G if their squares overlap and the overlap contains no witness. We present an output sensitive algorithm to compute a graph G based on its given witness unit square representation.

2012 ACM Subject Classification Mathematics of computing → Graph theory

Keywords and phrases proximity graphs, geometric intersection graphs, witness representation, unit square intersection graph, output sensitive algorithm, range searching

Digital Object Identifier 10.4230/LIPIcs.GD.2025.44

Category Poster Abstract

Funding *Soeren Terziadis*: funded by the NWO Gravitation project NETWORKS (no. 024.002.003.)

1 Introduction

Witness representations of graphs are a variant of geometric intersection representations of graphs, where we are given a set of convex objects and a set of points (*witnesses*). Every vertex is represented by one object and two vertices are connected if they intersect and their intersection contains no witness. Given a witness representation of n objects and m witnesses, we want to compute (*recover*) the graph it represents. Explicitly computing the intersection graph of all n objects which intersect at most $k \in O(n^2)$ times and checking for every intersection if it contains a witness requires $O(n^2 \cdot m)$ time. If the intersection graph is sparse, then the runtime can be reduced to $O(n \log m)$. However if there are $O(n^2)$ intersections between objects but the witness intersection graph has only $k \in O(n)$ edges, the naïve method would still require $O(n^2 \cdot m)$ time. Our goal is to reduce this to $\tilde{O}(n+m+k)$ by using an output sensitive algorithm. Towards this goal, we consider here the square version.

A witness unit square representation (wUSR) – also shown in Figure 1a) – is a tuple $(\mathcal{S}, \mathcal{W})$ where $\mathcal{S}$ is a set of n axis-aligned unit squares and $\mathcal{W}$ is a set of m point features (*witnesses*). The witness unit square graph $G(R) = \{\mathcal{S}, E\}$ is shown in Figure 1b). If $\{S_i, S_j\}$ intersect and their intersection contains a witness W, the edge $\{S_i, S_j\}$ is *blocked* by W.

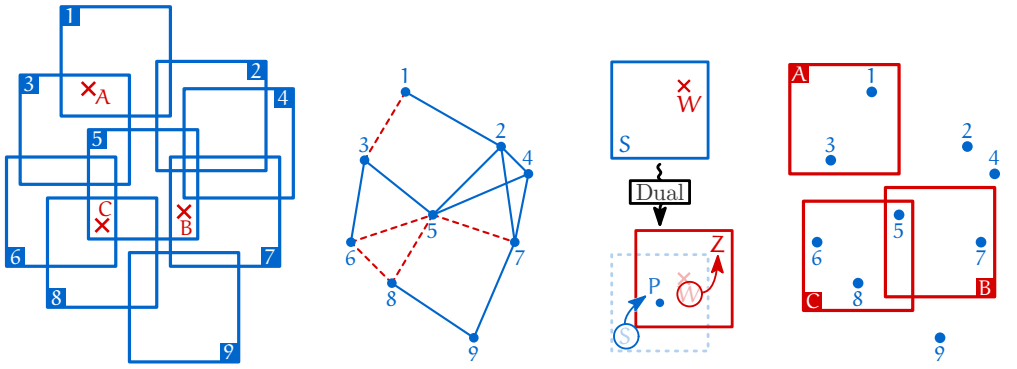
2 Initial Observations

To solve the problem we consider the dual of the representation, which represents any square in $\mathcal{S}$ with its center point and every witness with a unit square centered at the witness (Figure 1c). The *large square* B_i of S is a square of side length 2 centered at P_i.



© Maarten Löffler, Frank Staals, and Soeren Terziadis;
licensed under Creative Commons License CC-BY 4.0





(a) A wUSR $\mathcal{W}$ **(b)** Graph $G(\mathcal{W})$ **(c)** Duality **(d)** Dual representation

Figure 1 A wUSR $\mathcal{W}$ (a), its graph (b), the dual construction (c) and the dual of $\mathcal{W}$ (d).

▶ **Observation 1.** *Let $S_i, S_j \in \mathcal{S}$ be two intersecting squares and let $W_k \in \mathcal{W}$ be a witness such that $W_k \in S_i \cap S_j$. Then $P_i \in Z_k$ and $P_j \in Z_k$. Or in other words, if a witness is contained in the intersection of two squares, the dual points of both squares are contained in the dual square of the witness.*

▶ **Observation 2.** *Let $S_i, S_j \in \mathcal{S}$ be two intersecting squares. Then $P_j \in B_i$. Or in other words, if two squares intersect, then the dual point of one is in the large square of the other.*

▶ **Observation 3.** *Let $S_i \in \mathcal{S}$ be a squares and let $W_j \in \mathcal{W}$ be a witness such that $W_j \in S_i$. Then $Z_j \subset B_i$. Or in other words, if a square contains a witness, the large square of the square contains the dual square of the witness.*

Intuitively the previous observations state that everything relevant for a dual point of a square S is contained in its large square. We assume general position. The edges of $G(R)$ are partitioned into a set with positive slope (*right* edges) and negative slope (*left* edges). For the remainder of this paper we focus on right edges; left edges are handled symmetrically.

Consider the top-right quadrant $B^{\nearrow}$ of a large square B of S. For every right edge $\{S, S'\}$, with S lower than S', the dual point P' of S' is in $B^{\nearrow}$. For every witness W that is contained within S, we know by Observation 3 that it's dual square $Z \subset B$ and in particular the top-right corner of Z is contained in $B^{\nearrow}$. Let $\mathcal{T}$ be the set of top-right corners of witness dual squares that are contained in $B^{\nearrow}$. We say a point $A \in B^{\nearrow}$ dominates a point $A' \in B^{\nearrow}$ if A is higher and further right than A'.

▷ Claim 4. Let P, P' be two dual points of the squares S and S', respectively. Let $B^{\nearrow}$ be the top-right quadrant of the large square of S. The edge $\{P, P'\}$ is a right edge in $G(R)$, if and only if $P' \in B^{\nearrow}$ and there is no top right corner $T \in \mathcal{T} \cap B^{\nearrow}$, s.t., T dominates P'.

Proof. By Observation 2 S and S' overlap if and only if $P \in B^{\nearrow}$. By Observation 3 any witness contained in S has a dual square whose top right corner is contained in $B^{\nearrow}$, therefore no corners outside of $B^{\nearrow}$ have to be considered. Finally by Observation 1 any witness contained in both S and S' would have a top-right corner that dominates P' ◁

3 Recovery Algorithm

We build a two dimensional range tree on all points in the set $\mathcal{P}$, i.e., all dual points of squares in $\mathcal{S}$. Every secondary level node N corresponds to some region R_N, a (canonical) subset of points $P_N \subseteq \mathcal{P}$, and a subset $T_N \subseteq \mathcal{T}$. We order T_N to form a staircase. Let P'_N be

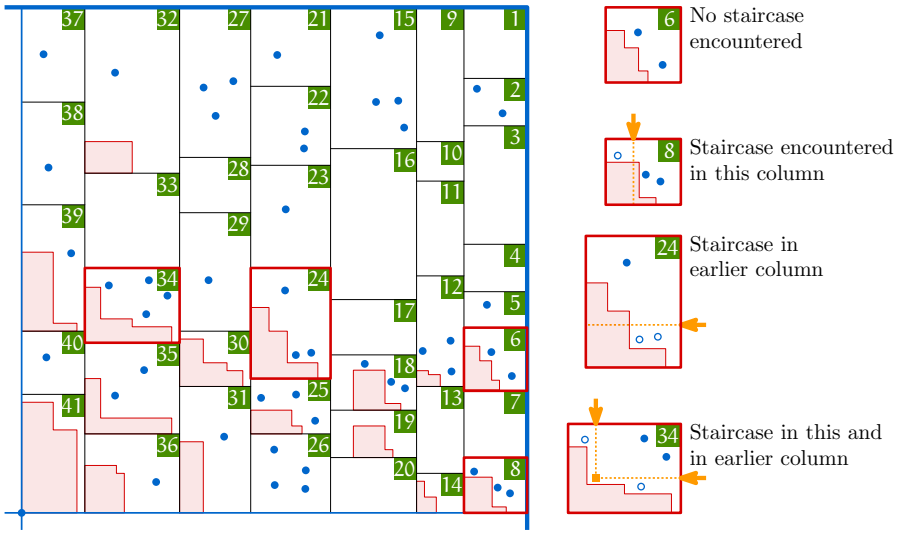


Figure 2 Query result as partition of query square with dual points (blue) and staircases of witness square corners (red). Cells are processed as numbered. Four cases for processing the cells are shown on the right.

the subset of P_N not dominated by a point in T_N, i.e. P'_N is the subset above the staircase. We store P'_N in a priority search tree [1] enabling efficient reporting of all points in P'_N, above and to the right of a query point. This can be done in $O((n+m)\log(n+m))$ time

For every point $P_i \in \mathcal{P}$ of a square $S_i \in \mathcal{S}$, we query this data-structure with a search window corresponding to $B_i^{\nearrow}$ of the large square of S_i in $O(\log^2 n)$.The query will report a number of subsets of $\mathcal{P}$. The returned sets can be interpreted as a partition of the search quadrant into columns, which are again subdivided into rows (forming *cells*); see Figure 2. The cells are processed column by column, right to left and top to bottom within a column. When processing these cells, we track $\hat{x}$ and $\hat{y}$, i.e., the highest x and y coordinates over all staircases and report the canonical subsets without points dominated by $(\hat{x}, \hat{y})$. Four cases of previous staircase encounters are distinguished (none, previously this column, previously in earlier column, previously this and earlier column); we refer again to Figure 2.

The last case (at most once per column, i.e., $O(\log n)$ times) requires an additional query in the priority search tree. Since the query square is subdivided into at most $O(\log^2 n)$ cells – let $\mathcal{C}$ be the set of all cells – in the query result of the range query, the entire time requirement for a single query plus processing of the cells is

$$O\left(\log^2 n\right) + O\left(\log n \cdot \log(n+m)\right) + O\left(\sum_{C\in\mathcal{C}} k_C\right) = O\left(\log^2(n+m) + k_i\right)$$

since $\sum_{C\in\mathcal{C}} k_C = k_i$, the number of reported edges. We need one query per dual point of a square in S and correctly compute all k right edges in time $O(n\log^2(n+m) + \sum_{i=1}^{n} k_i)$.

▶ **Theorem 5.** *Given a wUSR $\mathcal{W}$ with n unit squares and m witnesses, we can compute the implied graph $G(R)$ in $\tilde{O}(n+m+k)$ time, where k is the number of edges in $G(\mathcal{W})$.*

References

1 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: Algorithms and applications, 3rd Edition.* Springer, 2008. doi:10.1007/978-3-540-77974-2.

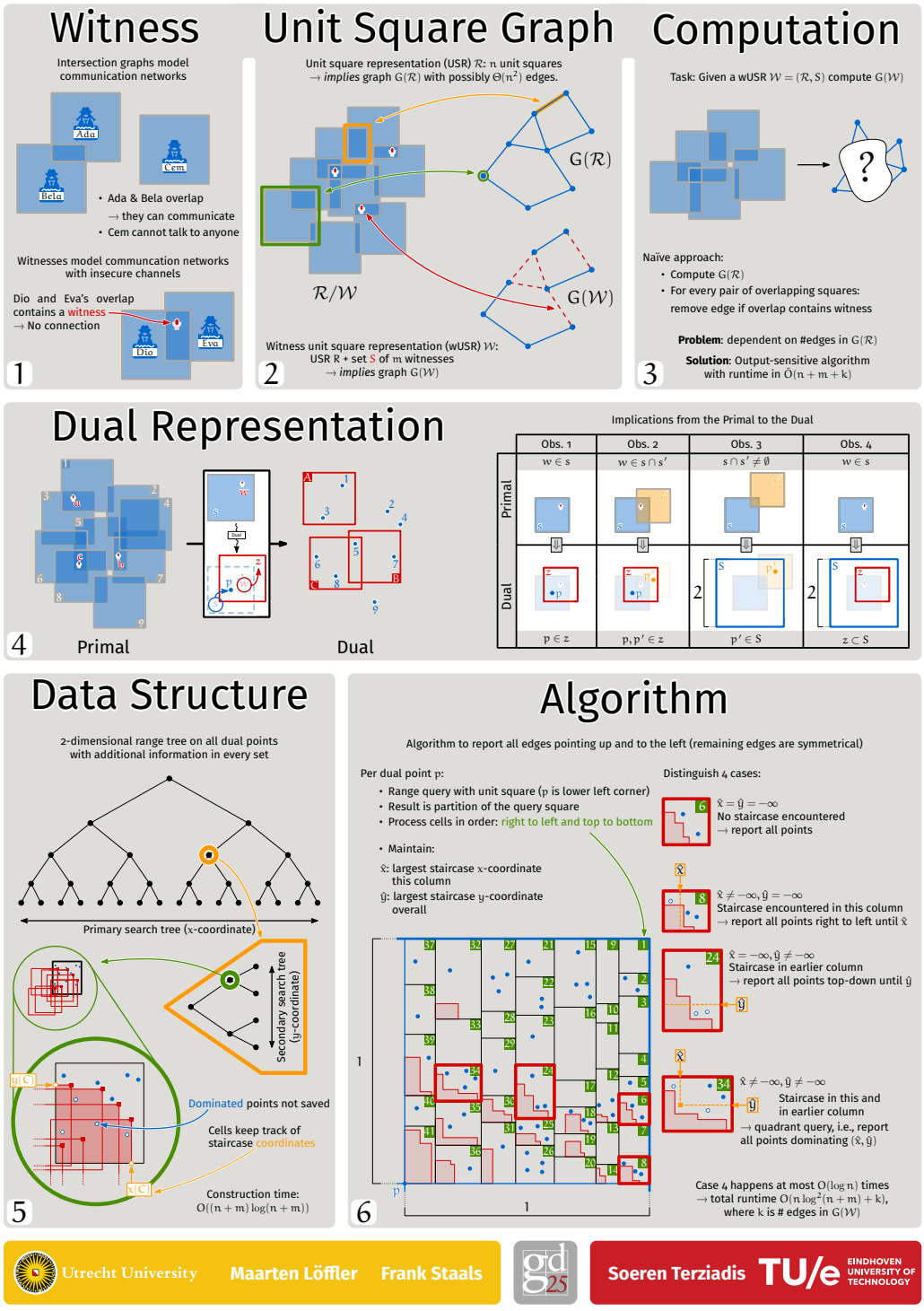


Figure 3 The poster.

TReView: Visualizing the European Union Transparency Register

Cristiano Bernardini
University of Perugia, Italy

Davide Campanelli
University of Perugia, Italy

Walter Didimo
University of Perugia, Italy

Luca Grilli
University of Perugia, Italy

Giuseppe Liotta
University of Perugia, Italy

Benedetto Ponti
University of Perugia, Italy

Abstract

We present TReView, the first visual analytics system for the exploration of the European Union (EU) Transparency Register, a large repository that aims to enhance transparency around lobbying activities within the EU, by enabling public oversight of meetings between lobbyists and EU officials.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Mathematics of computing → Graph theory; Theory of computation → Graph algorithms analysis

Keywords and phrases Transparency Registry, European Union, Graph Visualization, Interactive Visualization, Visual Analytics

Digital Object Identifier 10.4230/LIPIcs.GD.2025.45

Category Poster Abstract

Supplementary Material *Software (Source Code)*: `https://github.com/CriBerna1703/Transparency-Register-Application.git`

Funding *Cristiano Bernardini*: MUR PRIN 2022JSLZW, Assessing Conflict of Interests.
Walter Didimo: MUR PRIN P2022JSLZW, Assessing Conflict of Interests. MUR PRIN 2022TS4Y3N, EXPAND: scalable algorithms for EXPloratory Analyses of heterogeneous and dynamic Networked Data. MUR PON Proj. ARS01 00540, RASTA
Luca Grilli: MUR PRIN P2022JSLZW, Assessing Conflict of Interest.
Giuseppe Liotta: MUR PRIN 2022JSLZW, Assessing Conflict of Interests. MUR PRIN 2022TS4Y3N, EXPAND: scalable algorithms for EXPloratory Analyses of heterogeneous and dynamic Networked Data. MUR PON Proj. ARS01 00540, RASTA
Benedetto Ponti: MUR PRIN 2022JSLZW, Assessing Conflict of Interests.

Acknowledgements We are grateful to Francesco Annunziata for his help in collecting useful data.

1 Motivation and Contribution

The Transparency Register (TR) [9] is a public database established by the European Union (EU) in 2011 and continuously updated since then. It collects information about lobbyists – organizations, companies, and institutions – that seek to influence EU decision-making on specific topics or strategic areas of interest [2, 4, 5, 6]. However, despite the TR aims to



© Cristiano Bernardini, Davide Campanelli, Walter Didimo, Luca Grilli, Giuseppe Liotta, and Benedetto Ponti;
licensed under Creative Commons License CC-BY 4.0



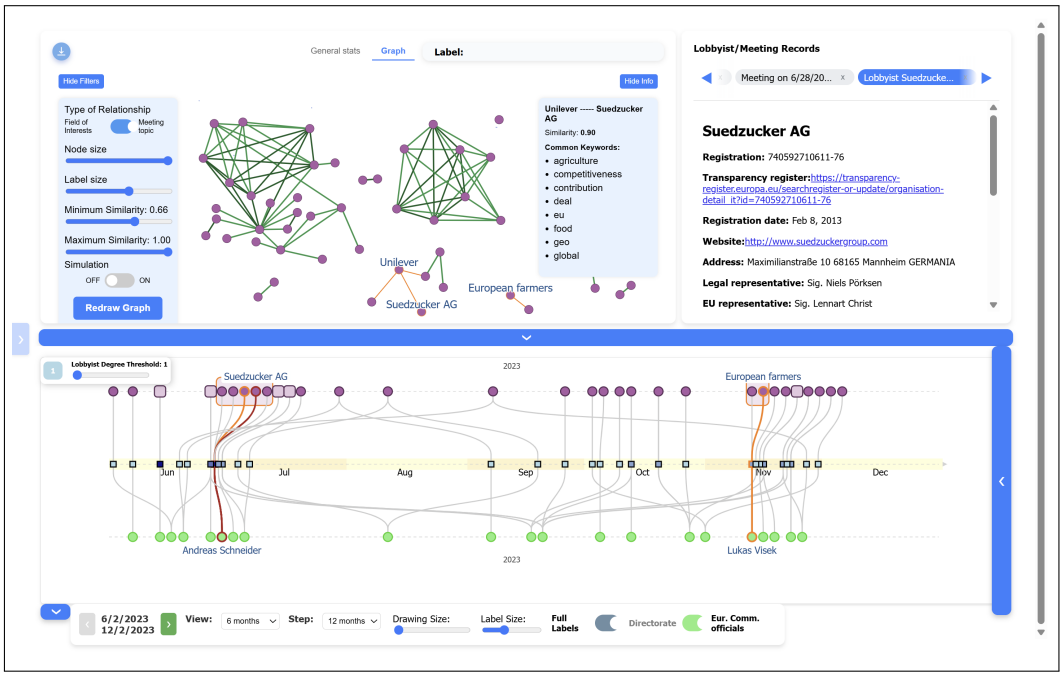


Figure 1 Snapshot of TReView's interface: (*i*) The Temporal view (bottom) reports the sequence of meetings between lobbyists and EU Commission officials; (*ii*) the Graph view (top-left) shows a similarity graph between lobbyists in terms of meeting topics; (*iii*) The Lobbyist/Meeting Records panel (top-right) reports details on lobbyists and meetings.

enhance transparency by enabling public oversight of meetings between lobbyists and EU officials, the TR's official website and related systems [3, 8] lack support for advanced data analysis. To address this limitation, we developed TReView, an interactive visual analytics system that employs multiple coordinated views to query and explore TR data from diverse perspectives TReView integrates a temporal view, a multilayer similarity graph view, and a flexible filtering subsystem, allowing users to focus on specific subsets of lobbyists, EU directorates, or topics of interest. The design of TReView has been driven by experts in the legal domain; its effectiveness is demonstrated through selected case studies.

2 System Design and Functionalities

The logical components, *data*, *users*, *tasks*, behind the design of TReView are the following [7].

- Data. We collected most of the information in the TReView database by a scraping procedure that massively queries the online web site of the TR[1]. We adopted this approach for two main reasons: (*i*) we could not find public APIs for accessing the TR; (*ii*) the open data made available by the EU[2] (as XML or Excel files) only contains partial information, not updated with the same frequency as the data in the web site.
- Users. TReView is designed for two main user categories: (*i*) *academic researchers* or *practitioners*, mainly in the fields of legal studies, political science, and sociology, interested in a better understanding of the dynamics of the lobbying activities behind the

[1] https://transparency-register.europa.eu/searchregister-or-update/search-register_en
[2] https://data.europa.eu/data/datasets/transparency-register?locale=en

decision-making process of the EU; (*ii*) *public officials, civil servants, journalists*, or *civil society activists*, interested in investigating specific organizations, persons, or topics, and in discovering potential conflicts of interest in the lobbying activities.

- Tasks. Following consultations with domain experts in legal studies and sociology, we identified a core set of three macro-tasks that TReView is designed to strongly support: T1 - Data Slicing and Filtering, to focus on specific time windows, lobbyists, organizations, Commission officials or directorates, and meeting topics. T2 - Temporal Analysis, to explore in detail the meetings between lobbyists and Commission officials along the temporal axis. T3 - Similarity Analysis, to recognize clusters of lobbyists that coordinate or behave similarly in terms of meeting topics or fields of interest.

User Interface. TReView's user interface is designed to support the tasks described above; see TReView in Figure 1 for a snapshot of the main views of the interface.

The Temporal View exploits a graph-based visualization on three layers. The central layer of the diagram corresponds to the time axis, with the indication of the years and the months. The meetings are displayed (as small squared nodes) along this axis at their corresponding dates. The top layer reports the lobbyists as purple disks, each connected to all meetings in which it is involved; to save space, maximal subsequences of consecutive degree-one lobbyists are grouped into a single (square-shaped) cluster node, which can be expanded to reveal its elements. The bottom layer reports the UE officials as green nodes, still connected to their corresponding meetings; the user may also choose to display directorates instead of EU officials. Clicking on a node, a bar chart of its meeting distribution over time is shown.

The Graph View depicts a similarity network in the classical node-link style: the nodes are the lobbyists involved in the analyzed time period and the edges are similarities between lobbyists, based either on common topics that characterize their meetings with the EU officials or on common fields of interest. The user can interact with this view to adjust the similarity threshold and recognize relevant clusters of lobbyists. The diagram is computed with the force-directed algorithm in the D3.js library [1], engineered to effectively handle the different connected components of the graph, which dynamically change throughout the user interaction. The system exploits fast NLP techniques for real-time extraction of meeting topics and similarity computation.

The Lobbyist/Meeting Records panel displays detailed information about specific lobbyists and meetings upon user's demand. At any time, a flexible Filter panel (not shown in Figure 1) can be used to focus on specific time periods, lobbyists, EU officials, and meeting's topics.

3 Case Studies and Demonstration Video

To assess the usefulness of TReView, we conducted two case studies, driven by experts in the legal domain. The first study analyzed the lobbying activities behind the Digital Markets and the Digital Services Acts from December 2020 to December 2022. In addition to offering a clear view of lobbying dynamics, the study revealed a concerning pattern that calls into question the principle of separation of powers – specifically, the divide between legislative and executive branches. The second case studies analyzed the the lobbying dynamics in the automotive sector since the launch of the EU Green Deal strategy on December 2019. It revealed that a high number of EU meetings were held with representative associations of the automotive sector rather than directly with key private companies in the field. Also, a surge in meetings in February 2025 allowed us to discover the promotion of a comprehensive EU strategy aimed to preserve industrial competitiveness and promote decarbonization at the same time. A short video on TReView is available at `https://youtu.be/lGq-h2c9GK0`.

References

1 Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, 17:2301–2309, 2011. doi:10.1109/TVCG.2011.185.

2 Raj Chari. New perspectives on lobbying regulation and reform: two generations of research. *Interest Groups & Advocacy*, 13:381–395, 2024.

3 Transparency International EU. EU Integrity Watch. https://www.integritywatch.eu/, 2025.

4 Nuria Font and Ixchel Pérez-Durán. Legislative transparency in the European Parliament: Disclosing legislators' meetings with interest groups. *JCMS*, 61(2):379–396, 2023.

5 Sofia Heikkonen. Transparency materialised: how registers can regulate access to documents? *Cambridge University Press*, 3(1), 2024.

6 Joan R. Martin and Irene A. Galcerà. Lobbying in the EU: prospects and challenges of the mandatory transparency register. *Front. Polit. Sci.*, 6, 2025. doi:10.3389/fpos.2024.1508017.

7 Silvia Miksch and Wolfgang Aigner. A matter of time: Applying a data-users-tasks design triangle to visual analytics of time-oriented data. *Comput. Graph.*, 38:286–290, 2014. doi:10.1016/J.CAG.2013.11.002.

8 Corporare Europe Observatory and LobbyControl. LobbyFacts.eu. https://www.lobbyfacts.eu/, 2025.

9 European Union. Trasparency register. https://transparency-register.europa.eu/, 2025.

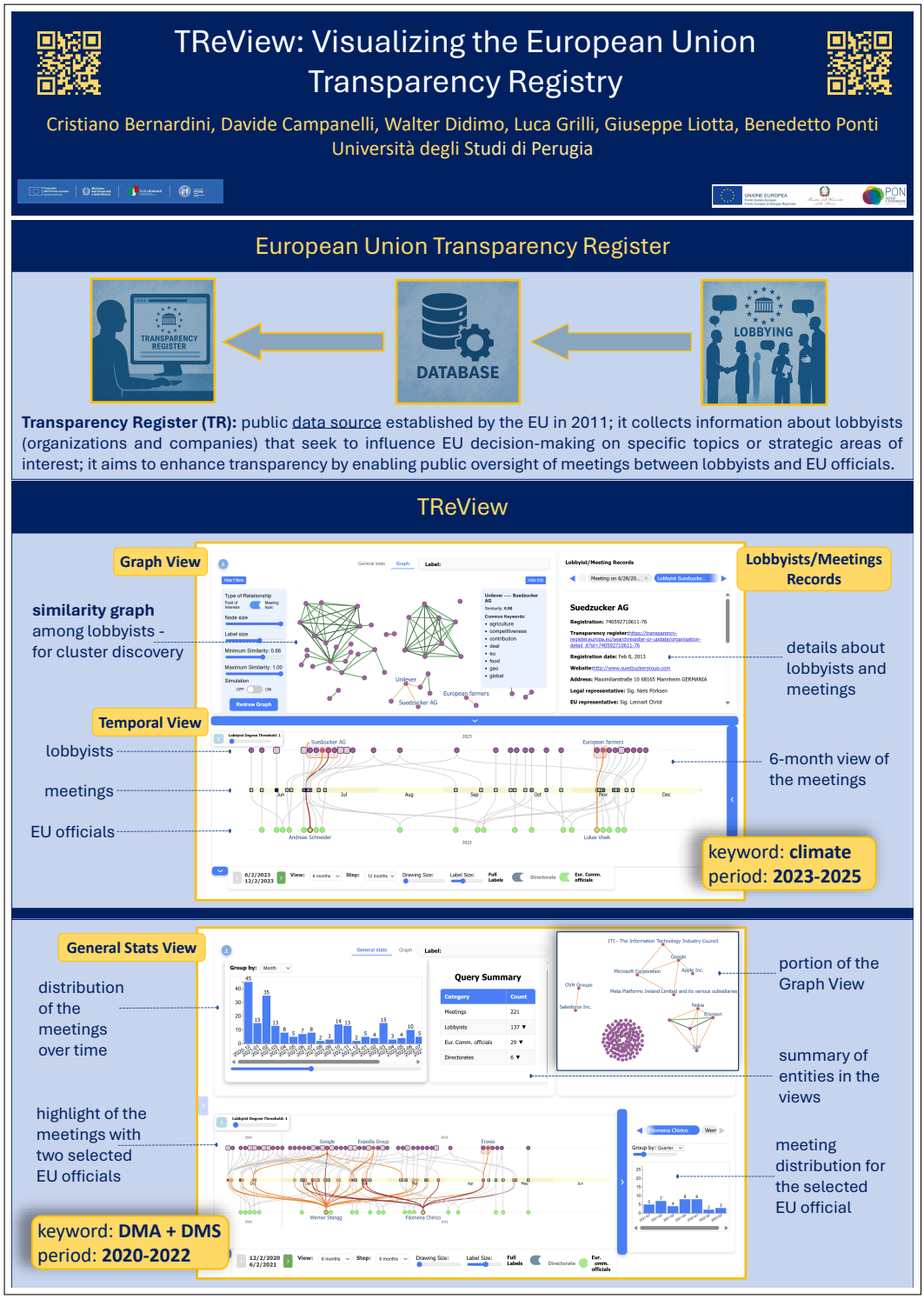


Figure 2 The poster.

Counting Triangulations of Fixed Cardinal Degrees

Erin Chambers
Department of Computer Science and Engineering, University of Notre Dame, IN, USA

Tim Ophelders
Department of Information and Computing Sciences, Utrecht University, The Netherlands
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Anna Schenfisch
Department of Mathematics, KTH Royal Institute of Technology, Stockholm, Sweden

Julia Sollberger
Department of Mathematics, Vrije Universiteit Amsterdam, The Netherlands

Abstract

A fixed set of vertices in the plane may have multiple planar straight-line triangulations in which the degree of each vertex is the same. As such, the degree information does not completely determine the triangulation. We show that even if we know, for each vertex, the number of neighbors in each of the four cardinal directions, the triangulation is not completely determined. We show that counting such triangulations is #P-hard via a reduction from #3-regular bipartite planar vertex cover.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Planar Triangulations, Degree Information, #P-Hardness

Digital Object Identifier 10.4230/LIPIcs.GD.2025.46

Category Poster Abstract

Funding *Erin Chambers*: Research supported in part by the National Science Foundation under award CCF-2444309.
Tim Ophelders: Partially supported by the Dutch Research Council (NWO) under project no. VI.Veni.212.260.
Anna Schenfisch: Supported by the Dutch Research Council (NWO) under project no. P21-13.

Paper Description

We consider *plane straight-line (PSL) graphs*: graphs whose vertices are points in the plane, and whose edges are interior-disjoint segments connecting the vertices at their endpoints. A PSL graph is *maximal* if adding any edge would result in a graph that is not a PSL graph. Given a PSL graph $G = (V, E)$ and $v \in V$, we write $\deg_{\mathcal{N}}(v)$ to mean the number of edges incident to v whose other endpoint lies above v in the y-direction ("to the north"). The *south-, east-, and west-degrees of* v, written $\deg_{\mathcal{S}}, \deg_{\mathcal{E}}, \deg_{\mathcal{W}} \colon V \to \mathbb{N}$ respectively, are defined analogously, and the four are collectively called the *cardinal degrees* of G. We consider only graphs with vertices at distinct x- and y-coordinates and in general position. We call $\sigma = (V, \deg_{\mathcal{N}}, \deg_{\mathcal{S}}, \deg_{\mathcal{E}}, \deg_{\mathcal{W}})$ a *cardinal signature* and call G a *realization* of σ if and

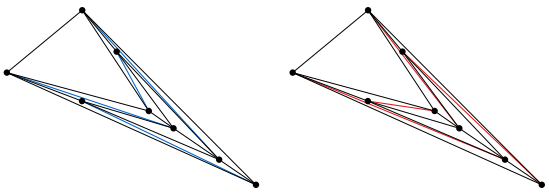


Figure 1 Maximal PSL graphs with identical cardinal signatures. Common edges are black.

© Erin Chambers, Tim Ophelders, Anna Schenfisch, and Julia Sollberger;
licensed under Creative Commons License CC-BY 4.0

only if G is a maximal PSL graph with cardinal signature σ. Although each graph has a unique cardinal signature, distinct graphs can have the same cardinal signature, even in the restricted setting of maximal PSL graphs; see Figure 1.

We therefore consider the problem of *counting* realizations of a cardinal degree signature, called #cardinal signature realization, and, in particular, we show this problem is #P-hard. We do so by a reduction from #3-regular bipartite planar vertex cover. However, we first reduce to an intermediary problem, namely, the #tiled noncrossing cycle-set problem, whose input is a *tiling*: a grid of *tiles*, where each tile has one of the *tile types* as in Figure 2, and are arranged to form a collection of red and blue cycles. A solution to this problem is a selection of noncrossing cycles. Figure 3 illustrates the connection between #tiled noncrossing cycle-set and #3-regular bipartite planar vertex cover.

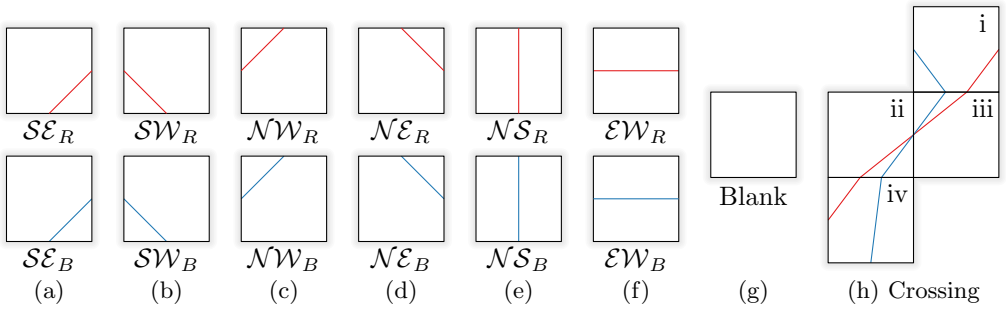


Figure 2 The 17 different tile types in the #tiled noncrossing cycle-set problem. In any tiling, the four tile types in (h) always appear together as pictured.

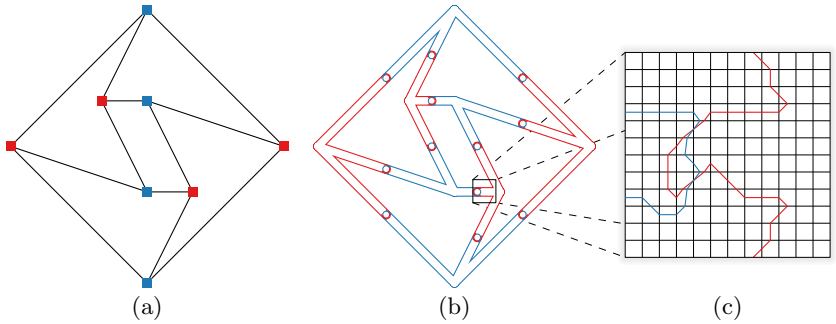


Figure 3 (a) An instance of #3-regular bipartite planar independent set. (b) A corresponding set of cycles in the plane. (c) Part of a corresponding tiling.

To connect our central problem to #tiled noncrossing cycle-set, we introduce *frame graphs*. A frame graph F is a grid-like pattern of convex chains of edges, forming roughly square faces called *frame cells*, corresponding to the tile structure of #tiled noncrossing cycle-set; see the black edges of Figure 4(c). Suppose a roughly horizontal path of F lies just below height h. This path is part of the boundary of the subgraph of F whose vertices lie below h. The sum of south degrees of all vertices below h tells us how many edges lie entirely in this halfspace. Using Euler's formula, we confirm that this lower subgraph is maximal PSL, i.e., the edges in this path must be present in every cardinal signature of F. Using a left-to-right argument for the nearly vertical paths tells us the realization of the cardinal signature of F is unique. By extension, any graph made by adding edges to F has a cardinal signature whose realizations all have the edges of F, i.e., these edges are *forced*. Thus, we include extra edges to a frame graph to form *gadgets*, corresponding to tile types; see Figure 4.

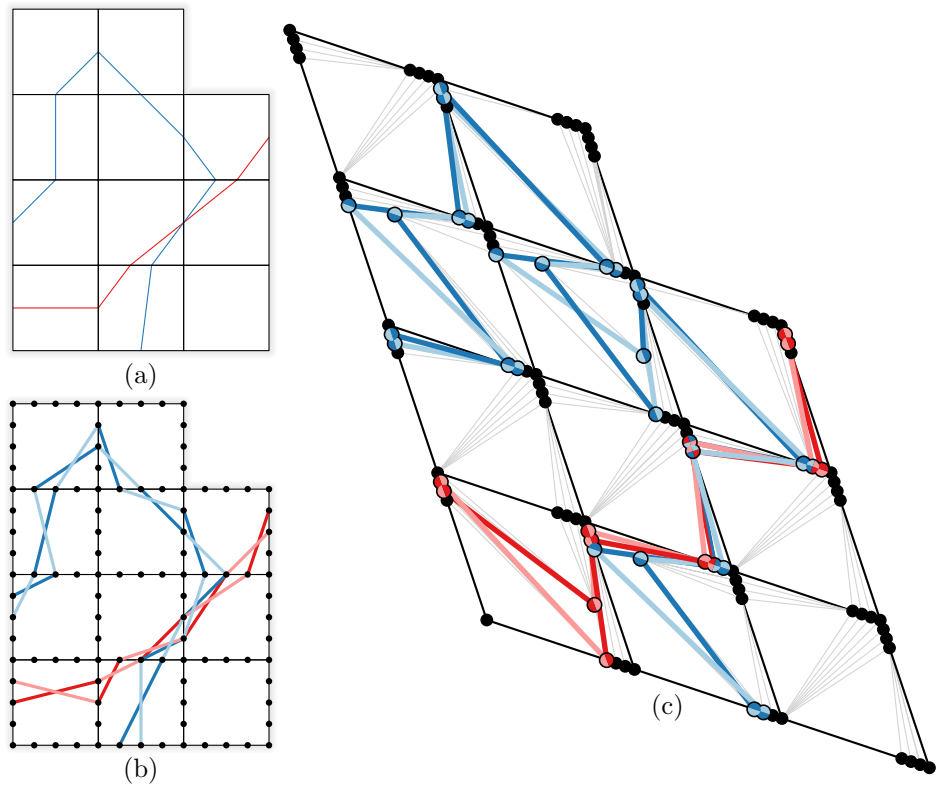


Figure 4 (a) Tile types for #tiled noncrossing cycle-set (up to exchanging colors). (b) Schematic representation of the gadgets of (c). (c) Corresponding gadgets in the same relative positions for #cardinal signature realization. Forced edges are grey and black.

We show that a frame graph triangulated by edges chosen from gadgets has a cardinal signature whose realizations are only the ones shown. We first establish our inductive step.

▶ **Lemma 1** (informal). *Let F be a frame graph triangulated by edges of gadgets. Let $\mathcal{G}$ be a gadget and suppose that the triangulation in F of the frame cell to the south of $\mathcal{G}$ is made by choosing light (dark) colored edges. Then the same choice of light (dark) colored edges triangulates $\mathcal{G}$ so that vertices on the southern boundary of $\mathcal{G}$ have the same cardinal degrees as they do in F. A similar statement applies to the west of $\mathcal{G}$.*

Since having a fixed triangulation to the south and west of a frame cell establishes the triangulation of the frame cell itself, and a consistent choice of light/dark leads to a consistent choice within the cell, we induct on the row and column of a frame cell to show the following.

▶ **Lemma 2** (informal). *All realizations of the cardinal signature of F correspond to triangulating frame cells using a subset of edges of the kind shown in Figure 4(c).*

We bijectively match each solution of a noncrossing tile selection to a realization of a cardinal signature σ; choosing (not choosing) a tile cycle corresponds to that portion of the triangulation made using light (dark) colored edges. We conclude that the number of cardinal signature realizations of σ is the same as the number of noncrossing tile selections of the corresponding cycle-set.

▶ **Theorem 3.** *#cardinal signature realization is #P-hard.*

Counting Triangulations of Fixed Cardinal Degrees

Erin Chambers
University of Notre Dame

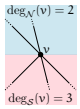
Tim Ophelders
Utrecht University
TU Eindhoven

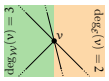
Anna Schenfisch
KTH

Julia Sollberger
VU Amsterdam

1

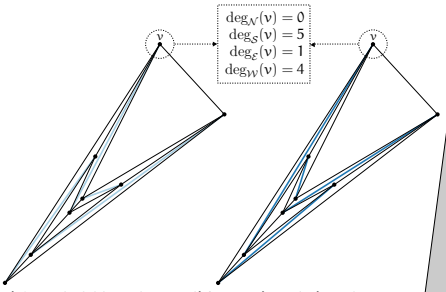
Cardinal degree of a vertex: counts of degree to the north, south, east, and west





Cardinal degree signature of a plane graph: position of vertices, and their cardinal degrees

Multiple graphs can have same cardinal signature!

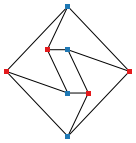


Light or dark blue edges → different triangulations, but every vertex keeps its cardinal degrees

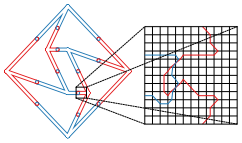
Computational question
How many realizations does a given cardinal degree signature have?

2

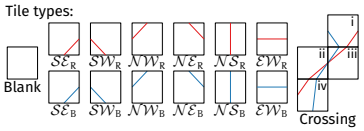
#P-hardness We reduce from the #P-hard problem of counting independent vertex sets in 3-regular planar bipartite graphs.



Reduce this to counting non-crossing subsets of cycles, where cycles are drawn on *tiles* of specific types.

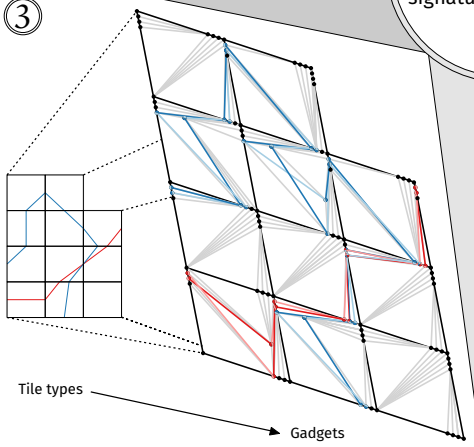


Tile types:



Counting graphs with a given cardinal degree signature is #P-hard.

3



Including a blue/red cycle in the set → analogous to choosing a triangulation using light blue/red in the corresponding places.

Not including a cycle → triangulations that use *dark* blue/red.

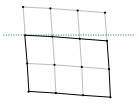
Any non-crossing cycle subset corresponds to a triangulation.

4

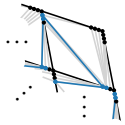
Why tiles and grid-like subgraphs? Why "tilted?"

- We know #edges below (left of) a horizontal line by summing degrees. By Euler's formula, subgraph below (to the left) must be convex
- Boundary of convex subgraph below/to the left is present in every realization

⇒ Grid edges must be present in any graph with the same cardinal degree signature.



Choosing light/dark red/blue to the south and west forces a cell to be triangulated with the same choices. (Otherwise cardinal degree signature changes).



Using gadgets to turn a tiled cycle set into a triangulation, non-crossing subsets of cycles are in bijection with triangulations with this cardinal signature!

Reconfigurations of Plane Caterpillars and Paths

Todor Antić
Department of Applied Mathematics, Faculty of Mathematics and Physics,
Charles University, Prague, Czech Republic

Guillermo Gamboa Quintero
Computer Science Institute of Charles University, Faculty of Mathematics and Physics,
Charles University, Prague, Czech Republic

Jelena Glišić
Department of Applied Mathematics, Faculty of Mathematics and Physics,
Charles University, Prague, Czech Republic

Abstract

Let S be a point set in the plane, and let $\mathcal{P}(S)$ and $\mathcal{C}(S)$ be the sets of all plane spanning paths and caterpillars on S. We study reconfiguration operations on $\mathcal{P}(S)$ and $\mathcal{C}(S)$. In particular, we prove that all of the commonly studied reconfigurations on plane spanning trees still yield connected reconfiguration graphs for caterpillars when S is in convex position. If S is in general position, we show that the rotation, compatible flip and flip graphs of $\mathcal{C}(S)$ are connected while the slide graph is sometimes disconnected, but always has a component of size $\frac{1}{4}(3^n - 1)$. We then study sizes of connected components in reconfiguration graphs of plane spanning paths. In this direction, we show that no component of size at most 7 can exist in the flip graph on $\mathcal{P}(S)$.

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

Keywords and phrases reconfiguration graph, caterpillar, path, geometric graph

Digital Object Identifier 10.4230/LIPIcs.GD.2025.47

Category Poster Abstract

Related Version *Full Version*: `https://arxiv.org/abs/2410.07419`

Supplementary Material
Software (Source code): `https://github.com/jelena-glisic/Caterpillar-Slides` [6]
archived at `swh:1:dir:552a6b334147879c561c75db5d9b586bf6c23aa2`

Funding *Todor Antić*: supported by grant no. 23-04949X of the Czech Science Foundation (GAČR) and by SVV–2023–260699.
Guillermo Gamboa Quintero: supported by the GAČR grant 22-17398S and the Charles University project PRIMUS/24/SCI/012.
Jelena Glišić: supported by grant no. 23-04949X of the Czech Science Foundation (GAČR) and by SVV–2023–260699.

Acknowledgements We would like to thank Jan Kynčl and Pavel Valtr for proposing the problems and for all of the helpful discussions during the combinatorial problems seminar at Charles University. We also thank Daniel Perz for pointing out the existence of an isolated vertex in $G_{\mathcal{C}}^{\text{slide}}(S)$, for his help in finding literature for this paper, and helpful comments.

1 Introduction

Given a set of structures C, and a reconfiguration operation that transforms one object in C to another, the *reconfiguration graph* is a graph with vertex set C in which two vertices form an edge if one can be transformed into the other using a single reconfiguration operation. Often, in computer science, objects are solutions to a problem and reconfigurations are local


© Todor Antić, Guillermo Gamboa Quintero, and Jelena Glišić;
licensed under Creative Commons License CC-BY 4.0

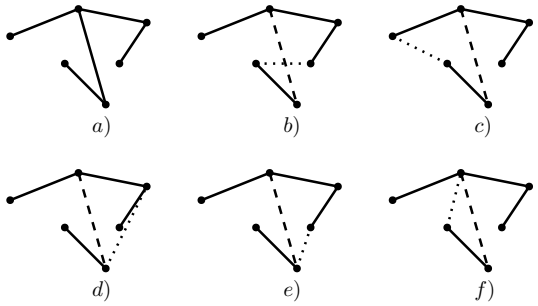


Figure 1 a) A plane spanning tree. Replacing the dashed line with the dotted line corresponds to: b) a flip, c) a compatible flip, d) a rotation, e) an empty triangle rotation and f) a slide.

changes that transform one solution into another. Then, to understand the solution space, it is important to study the reconfiguration graph. For an introduction to the topic, see [14].

Given a point set S in the plane, a *plane spanning tree* on S is a spanning tree of S whose edges are straight line segments that do not cross. Let $\mathcal{T}(S)$ be the set of all plane spanning trees on S. We consider five reconfigurations on $\mathcal{T}(S)$ (see Figure 1). For the following, we are given $T_1 = (S, E_1), T_2 = (S, E_2) \in \mathcal{T}(S)$, and we say that:

1. T_1 and T_2 are connected by a *flip* if $E_2 = E_1 \setminus \{e\} \cup \{f\}$ for some edges e, f.
2. T_1 and T_2 are connected by a *compatible flip* if $E_2 = E_1 \setminus \{e\} \cup \{f\}$ for some edges e, f which do not cross.
3. T_1 and T_2 are connected by a *rotation* if $E_2 = E_1 \setminus \{e\} \cup \{f\}$ for some edges e, f which share an endpoint.
4. T_1 and T_2 are connected by an *empty triangle rotation* if $E_2 = E_1 \setminus \{e\} \cup \{f\}$ for some edges e, f which share an endpoint and the triangle spanned by their endpoints is empty.
5. T_1 and T_2 are connected by a *slide* if $E_2 = E_1 \setminus \{e\} \cup \{f\}$ for some edges e, f which are as in 4. and if $e = ab$ and $f = ac$ then $bc \in E_1 \cap E_2$.

Note that every slide is an empty triangle rotation, every empty triangle rotation is a rotation, and so on. This hierarchy will be useful when studying the structural properties of the corresponding reconfiguration graphs.

2 Background

The reconfiguration graphs associated with the operations described above have been a topic of interest for a long time with many results appearing through the years. These results have concerned connectivity [1, 7, 13], lower and upper bounds on the diameter [2, 9, 10], etc.

It is known that the reconfiguration graph of plane spanning trees is connected even in the most restrictive case when the reconfigurations are slides [1]. In the case of slides, a tight $\Theta(n^2)$ bound on the diameter is shown in [4]. For the empty triangle rotations, an upper bound of $O(n \log n)$ on the diameter was shown in [13], while for the remaining reconfigurations, a linear upper bound is known [7]. In [2], an upper bound of $2n - 3$ is shown for flip graphs. The best known lower bound on the diameter of the reconfiguration graphs is $\frac{14}{9}n - O(1)$, as per [8].

However, there has been little research on induced subgraphs of these reconfiguration graphs. The only such subgraph that has been explored is the one induced by plane spanning paths. For a set of n points in convex position, it is known that the flip graph is connected [5] and that it has diameter $2n - 5$ for $n \in \{3, 4\}$ and $2n - 6$ for $n \geq 5$ [11]. The flip graph of

point sets with at most two convex layers is connected [12], as is the flip graph of generalized double circles [3]. For point sets in general position, connectivity was first conjectured by Akl, Islam and Meijer [5] 18 years ago but still remains unsolved.

3 Our contribution

We further the study of induced subgraphs of reconfiguration graphs of plane spanning trees by exploring reconfigurations of plane spanning caterpillars, and by expanding on the topic of reconfigurations of plane spanning paths. A *caterpillar* is a tree in which all non-leaf vertices form a path. We call this path the *spine* of the caterpillar. A *plane spanning caterpillar* of a point set S is a plane spanning tree of S which is a caterpillar. For a set S, we will denote by $\mathcal{C}(S)$ the set of all plane spanning caterpillars on S. We will denote the reconfiguration graphs on $\mathcal{C}(S)$ by $G_{\mathcal{C}}^{\text{flip}}(S)$, $G_{\mathcal{C}}^{\text{comp-flip}}(S)$, $G_{\mathcal{C}}^{\text{rot}}(S)$, $G_{\mathcal{C}}^{\text{emp-rot}}(S)$, and $G_{\mathcal{C}}^{\text{slide}}(S)$. First, we focus on the case when S is in convex position. We show that the *slide graph* $G_{\mathcal{C}}^{\text{slide}}$ is connected, which implies that all of the reconfiguration graphs are connected in this case. For compatible flips, we prove a stronger bound for the diameter.

▶ **Theorem 1.** *Let S be a set of $n \geq 3$ points in convex position in the plane. Then, the graph $G_{\mathcal{C}}^{slide}(S)$ is connected with diameter at most $3n - 8$.*

▶ **Theorem 2.** *Let S be a set of $n \geq 3$ points in convex position in the plane. Then, the graph $G_{\mathcal{C}}^{comp\text{-}flip}(S)$ is connected with diameter at most $2n - 5$.*

Next, we consider the case when S is a point set in general position. For a caterpillar $C \in \mathcal{C}(S)$, and consecutive spine vertices $v_i, \ldots, v_j$ of C, we write $S_{i,j}$ for the point set consisting of the spine vertices and all of the leaves attached to them. We call $C \in \mathcal{C}(S)$ with spine $v_1, v_2, \ldots, v_k$ a *well-separated* caterpillar if for each $i \geq 1$, the convex hull of $S_{1,i}$ is disjoint from the rest of S. We can prove that all caterpillars in this class are mutually connected in $G_{\mathcal{C}}^{\text{slide}}(S)$. This large connected component is significant, as we can also show that $G_{\mathcal{C}}^{\text{slide}}(S)$ is disconnected. On the other hand, we can prove that the rotation graph $G_{\mathcal{C}}^{\text{rot}}(S)$ is connected. We leave open the question of the connectivity of $G_{\mathcal{C}}^{\text{emp-rot}}$.

▶ **Theorem 3.** *Any two well-separated caterpillars are connected in $G_{\mathcal{C}}^{slide}(S)$.*

▶ **Theorem 4.** *The graph $G_{\mathcal{C}}^{slide}(S)$ is connected for every set S of n points in the plane if $n \leq 7$. If $n \geq 8$, there exists a set S of n points such that $G_{\mathcal{C}}^{slide}(S)$ has isolated vertices.*

▶ **Proposition 5.** *The graph $G_{\mathcal{C}}^{rot}(S)$ is connected.*

Lastly, we focus on connected components of the reconfiguration graph of plane spanning paths. Given a set of points in general position S, we will call the corresponding *flip graph of plane spanning paths* $G_{\mathcal{P}}(S)$. Currently, the main open problem is deciding whether $G_{\mathcal{P}}(S)$ is connected. Here we find a large connected component consisting of what we call *generalized peeling paths*. We note that Theorem 6 was independently discovered by Kleist, Kramer, and Rieck [12]. We use the number of these paths to prove that there are at least $\frac{1}{4}(3^n - 1)$ well-separated caterpillars on S. Finally, we investigate the minimal size of components in $G_{\mathcal{P}}(S)$.

▶ **Theorem 6.** *Let S be a set of n points in general position. Then $G_{\mathcal{P}}(S)$ contains a connected component of size $2^{\Omega(n)}$.*

▶ **Theorem 7.** *Let S be a point set of $n \geq 5$ points in general position. Then, $G_{\mathcal{P}}(S)$ contains no connected component of size at most 7.*

References

1 Oswin Aichholzer, Franz Aurenhammer, and Ferran Hurtado. Sequences of spanning trees and a fixed tree theorem. *Computational Geometry*, 21(1-2):3–20, 2002. doi:10.1016/S0925-7721(01)00042-6.

2 Oswin Aichholzer, Brad Ballinger, Therese Biedl, Mirela Damian, Erik D. Demaine, Adam Hesterberg, Matias Korman, Anna Lubiw, Jayson Lynch, Josef Tkadlec, and et al. Reconfiguration of non-crossing spanning trees. *Journal of Computational Geometry*, 15(1):224–253, December 2024. doi:10.20382/jocg.v15i1a9.

3 Oswin Aichholzer, Kristin Knorr, Wolfgang Mulzer, Johannes Obenaus, Rosna Paul, and Birgit Vogtenhuber. Flipping plane spanning paths. In *WALCOM*, volume 13973 of *Lecture Notes in Computer Science*, pages 49–60. Springer, 2023. doi:10.1007/978-3-031-27051-2_5.

4 Oswin Aichholzer and Klaus Reinhardt. A quadratic distance bound on sliding between crossing-free spanning trees. *Computational Geometry*, 37(3):155–161, 2007. doi:10.1016/j.comgeo.2004.12.010.

5 Selim G. Akl, Kamrul Islam, and Henk Meijer. On planar path transformation. *Information Processing Letters*, 104(2):59–64, 2007. doi:10.1016/j.ipl.2007.05.009.

6 Todor Antić, Guillermo Gamboa Quintero, and Jelena Glišić. jelena-glisic/Caterpillar-Slides. Software, swhId: swh:1:dir:552a6b334147879c561c75db5d9b586bf6c23aa2 (visited on 2025-09-12). URL: https://github.com/jelena-glisic/Caterpillar-Slides, doi:10.4230/artifacts.24692.

7 David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21–46, 1996. doi:10.1016/0166-218X(95)00026-N.

8 Håvard Bakke Bjerkevik, Linda Kleist, Torsten Ueckerdt, and Birgit Vogtenhuber. *Flipping Non-Crossing Spanning Trees*, pages 2313–2325. Society for Industrial and Applied Mathematics, 2025. doi:10.1137/1.9781611978322.77.

9 Nicolas Bousquet, Lucas de Meyer, Théo Pierron, and Alexandra Wesolek. Reconfiguration of Plane Trees in Convex Geometric Graphs. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th International Symposium on Computational Geometry (SoCG 2024)*, volume 293 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:17, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2024.22.

10 Nicolas Bousquet, Valentin Gledel, Jonathan Narboni, and Théo Pierron. A note on the flip distance between non-crossing spanning trees. *Computating in Geometry and Topology*, 2(1):8:1–8:7, 2023. doi:10.57717/cgt.v2i1.36.

11 Jou-Ming Chang and Ro-Yu Wu. On the diameter of geometric path graphs of points in convex position. *Information Processing Letters*, 109(8):409–413, 2009. doi:10.1016/J.IPL.2008.12.017.

12 Linda Kleist, Peter Kramer, and Christian Rieck. On the connectivity of the flip graph of plane spanning paths. In Daniel Kráľ and Martin Milanič, editors, *Graph-Theoretic Concepts in Computer Science*, pages 327–342, Cham, 2025. Springer Nature Switzerland. doi:10.1007/978-3-031-75409-8_23.

13 Torrie L. Nichols, Alexander Pilz, Csaba D. Tóth, and Ahad N. Zehmakan. Transition operations over plane trees. *Discrete Mathematics*, 343(8):111929, 2020. doi:10.1016/j.disc.2020.111929.

14 Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4), 2018. doi:10.3390/a11040052.

Reconfigurations of Plane Caterpillars (and Paths)
Todor Antić
Guillermo Gamboa Quintero
Jelena Glišić
CHARLES UNIVERSITY
Definitions
Reconfiguration = replacing the red edge by a blue edge
flip
slide
rotation
edges can't cross!
edges must share a vertex!
△ must be empty!
adjacent to both edges
• S . . . a pointset
• n . . . size of S
• $\mathcal{C}'(S)$ . . . plane spanning caterpillars
• $G_{\mathcal{C}}^{\text{flip}}(S)$ . . . flip graph
• $G_{\mathcal{C}}^{\text{comp-flip}}(S)$ . . . comaptible flip graph
• $G_{\mathcal{C}}^{\text{rot}}(S)$ . . . rotation graph
• $G_{\mathcal{C}}^{\text{emp-rot}}(S)$ . . . empty △ rotation graph
• $G_{\mathcal{C}}^{\text{slide}}(S)$ . . . slide graph
Reconfigurations form a hierarchy!
Convex position
Slides
Thm: S convex $\Longrightarrow$ $G_{\mathcal{C}}^{\text{slide}}(S)$ is connected with diameter at most $3n-8$.
Idea: Slide to a star.
Each edge is moved at most once $\Longrightarrow$ we use $\leq n-3$ slides!
Total slides: $(n-3)+(n-2)+(n-3)=3n-8$
Compatible flips
Thm: S convex $\Longrightarrow$ $G_{\mathcal{C}}^{\text{comp-flip}}(S)$ is connected with diameter at most $2n-5$.
Idea: Flip to a path along the convex hull.
Each edge is moved at most once and egdes on the convex hull are never moved $\Longrightarrow$ we use $\leq n-3$ compatible flips!
Total compatible flips: $(n-3)+1+(n-3)=2n-5$
General position
Slides
Thm: S in general position $\Longrightarrow$ If $n \leq 7$ then $G_{\mathcal{C}}^{\text{slide}}(S)$ is connected. If $n \geq 8$ then $G_{\mathcal{C}}^{\text{slide}}(S)$ can be disconnected.
Lem: S in general position and $u, v \in S$ $\Longrightarrow$ any double star with centers u and v is connected to a star at u (or v) in $G_{\mathcal{C}}^{\text{slide}}$.
Thm: S in general position $\Longrightarrow$ $G_{\mathcal{C}}^{\text{slide}}(S)$ contains a connected component of size $\geq \frac{1}{4}(3^n-1)$.
I'm in the component!
well-separated caterpillar
not a well-separated caterpillar
Rotations
Thm: S in general position $\Longrightarrow$ $G_{\mathcal{C}}^{\text{rot}}(S)$ is connected.
Idea: Induce on spine length
Shortening the spine:
Apply lemma in the bounded region!
Open problems
• Improve diameter bounds
• Is $G_{\mathcal{C}}^{\text{emp-rot}}(S)$ connected for S in general position?
• Investigate happy edge properties
• Complexity related questions
A new result about paths
Thm: S in general position $\Longrightarrow$ the flip graph of plane spanning paths on S does not contain a connected component of size ≤ 7.
Check out the Arxiv version!

Drawing Trees and Cacti with Integer Edge Lengths on a Polynomial-Size Grid

Henry Förster
Technical University of Munich, Germany

Stephen Kobourov
Technical University of Munich, Germany

Jacob Miller
Technical University of Munich, Germany

Johannes Zink
Technical University of Munich, Germany

Abstract

A strengthened version of Harborth's well-known conjecture – known as Kleber's conjecture – states that every planar graph admits a planar straight-line drawing where every edge has integer length and each vertex is restricted to the integer grid. Positive results for Kleber's conjecture are known for planar 3-regular graphs, for planar graphs that have maximum degree 4, and for planar 3-trees. However, all but one of the existing results are existential and do not provide bounds on the required grid size. We provide polynomial-time algorithms for computing crossing-free straight-line drawings of trees and cactus graphs with integer edge lengths and integer vertex position on polynomial-size integer grids. We also give an historic overview of planar straight-line graph drawing results.

2012 ACM Subject Classification Mathematics of computing → Trees; Mathematics of computing → Graph algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases Harborth's conjecture, tree drawings, cactus drawings, grid drawings

Digital Object Identifier 10.4230/LIPIcs.GD.2025.48

Category Poster Abstract

Related Version *Full Version*: `https://arxiv.org/abs/2509.04168`

1 History of planar straight-line graph drawings

Wagner in 1936 [17], Fáry in 1948 [7], and Stein in 1951 [14] proved independently that every planar graph has a crossings-free drawing with edges drawn by straight-line segments. After these first existential results, algorithms that do create such drawings were described by Tutte in 1963 [16], Chiba, Yamanouchi, and Nishizeki in 1984 [5], and Read in 1986 [12]. In the generated drawings, the ratio between longest and shortest edge can be exponential in the size of the graph. For readability, it helps to have vertices at integer coordinates on a polynomial-size grid. In 1990, two algorithms generating such drawings appeared. De Fraysseix, Pach and Polack [6] use canonical orders whereas Schnyder's algorithm [13] uses a decomposition of the graph into three trees from which barycentric coordinates can be computed. For n vertices, both algorithms provide a straight-line drawing on the $\mathcal{O}(n) \times \mathcal{O}(n)$ grid.

Can we also ensure that all edge lengths are integers? These two algorithms [6, 13] can (and do) produce irrational edge lengths. Harborth's conjecture [9, 10] asks whether every planar graph has a crossings-free realization with straight-line segments of integer lengths, referred to integral Fáry embeddings [4]. A stronger version of Harborth's conjecture, called Kleber's conjecture, asks whether an integral Fáry embedding exists with all vertices on the integer grid [11]. We refer to these drawings as *truly integral Fáry embeddings*.


© Henry Förster, Stephen Kobourov, Jacob Miller, and Johannes Zink;
licensed under Creative Commons License CC-BY 4.0



Sun [15] uses rigidity matrices to prove that all planar (2, 3)-sparse graphs admit integral Fáry embeddings. A graph G is (a, b)-sparse if every subgraph G' of G has at most $a|V(G')|-b$ edges. (2, 3)-sparse graphs include series-parallel, outerplanar, and planar bipartite graphs. In 2024, Chang and Sun [4] show this result also for all planar 4-regular graphs.

Geelen, Guo, and McKinnon [8] show that every planar max-degree-3 graph admits a truly integral Fáry embedding. Biedl [3] observes that this technique can be applied to all planar (2, 1)-sparse graphs, which include all max-degree-4 graphs that are non-4-regular. Independently, Benediktovich [1] shows this result for max-degree-4 graphs that are non-4-regular and for planar 3-trees. The vertices have rational coordinates, thus, can be scaled to integers. However, they make use of a theorem by Berry [2] to position vertices on a rational point in an ε-disk around a real point, which doesn't provide guarantees on the distance towards an integral point. If the drawing is scaled to provide a truly integral Fáry embedding, no bound on the area can be provided. To the best of our knowledge, the algorithm by Biedl [3] that produces quadratic truly integral Fáry embeddings for 3-regular planar graphs is the only algorithm in the literature that provides a polynomial upper bound on the area.

2 Truly integral Fáry embeddings of trees and cacti on a poly-size grid

We show that trees and cacti admit truly integral Fáry embeddings on the $\mathcal{O}(n^2) \times \mathcal{O}(n^2)$ and $\mathcal{O}(n^3) \times \mathcal{O}(n^3)$ integer grids, respectively. For binary trees, $\mathcal{O}(n^2)$ follows from [3]; for constant-depth trees like stars, $\mathcal{O}(n^2)$ follows from Theorem 1. To ensure integer edge lengths on an integer grid, we are restricted to using Pythagorean triples. Pythagorean triples are all triples of integers a, b, c which satisfy $a^2 + b^2 = c^2$. A Pythagorean triple is primitive if a, b, and c are co-prime. The first k primitive Pythagorean triples $\mathcal{P}_k$ can be computed in $\mathcal{O}(k^{3/2})$ time. We denote by $\mathcal{P}_k^\circ$ the angle-sorted first k primitive Pythagorean triples.

Let $T = (V, E)$ be a (rooted) tree, which has n vertices, ℓ leaves, and depth d. Denote the root by r; if no root is given, pick r arbitrarily. If no rotation system is given, choose it arbitrarily. For a vertex $v \in V$, we denote by T_v the subtree of T rooted at vertex v and by $L(T_v)$ the number of leaves in T_v. Place r at $(0, 0)$. Assign the first child v_1 of r the first $L(T_{v_1})$ primitive Pythagorean triples from $\mathcal{P}_\ell^\circ$, assign the second child v_2 of r the next $L(T_{v_2})$ primitive Pythagorean triples from $\mathcal{P}_\ell^\circ$, etc. Draw each child v of r at the coordinates (x, y) where (x, y, z) is the first primitive Pythagorean triple assigned to v. For each child v of r, for which v is not a leaf of T, we draw the subtree T_v recursively: in the recursive call, we assume that the coordinate system is centered at the position of v and we use exactly the primitive Pythagorean triples assigned to v; see Figure 1.

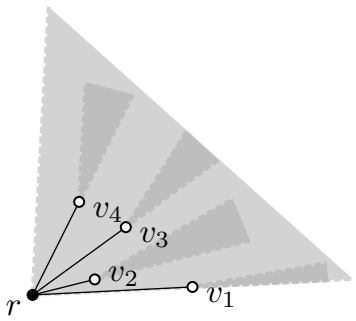


Figure 1 Schematization of our recursive algorithm for drawing trees. Vertex r is placed at $(0, 0)$ and its subtree is drawn inside the light-gray cone; v_i is drawn at its first assigned triple.

Correctness follows from the fact that we use distinct slopes (primitive Pythagorean triples) in distinct subtrees that are sorted such that they fan out. The realization as a linear-time algorithm is straight-forward, however, we add $\mathcal{O}(\ell^{3/2})$ additional running time for computing primitive Pythagorean triples, which needs to be done only once.

▶ **Theorem 1.** *Let $T = (V, E)$ be a (rooted) tree, which has n vertices, ℓ leaves, and depth d. There is a truly integral Fáry embedding on a grid of size $\frac{2\pi^2}{3}\ell d \times \frac{2\pi^2}{3}\ell d \subseteq \mathcal{O}(n^2) \times \mathcal{O}(n^2)$, which can be found in $\mathcal{O}(n + \ell^{3/2})$ time.*

With a special handling of cycles, in particular triangles, we can extend the algorithm from trees to cacti and obtain the following result.

▶ **Theorem 2.** *Let $C = (V, E)$ be a (rooted) cactus, which has n vertices, ℓ leaves, diameter d, o cycles out of which δ are triangles. There is a truly integral Fáry embedding on a grid of size $\left(\frac{2\pi^2}{3}(d+o)(\ell+2o) + \delta \cdot 2\left(\frac{\pi^2}{3}\cdot(\ell+2o)\right)^2\right) \times \left(\frac{2\pi^2}{3}(d+o)(\ell+2o) + \delta \cdot 2\left(\frac{\pi^2}{3}\cdot(\ell+2o)\right)^2\right)$ $\subseteq \mathcal{O}(n^3) \times \mathcal{O}(n^3)$, which can be found in $\mathcal{O}(n + (\ell+2o)^{3/2})$ time.*

References

1. Vladimir I. Benediktovich. On rational approximation of a geometric graph. *Discret. Math.*, 313(20):2061–2064, 2013. doi:10.1016/J.DISC.2013.06.018.
2. T.G. Berry. Points at rational distance from the vertices of a triangle. *Acta Arithmetica*, LXII(4):391–398, 1992. doi:10.4064/aa-62-4-391-398.
3. Therese Biedl. Drawing some planar graphs with integer edge-lengths. In *Proc. CCCG 2011*, 2011. URL: http://www.cccg.ca/proceedings/2011/papers/paper36.pdf.
4. Daniel J. Chang and Timothy Sun. Harborth's conjecture for 4-regular planar graphs. In *Proc. GD 2024*, 2024. doi:10.4230/LIPICS.GD.2024.38.
5. Norishige Chiba, Tadashi Yamanouchi, and Takao Nishizeki. Linear algorithms for convex drawings of planar graphs. *Progress in graph theory*, pages 153–173, 1984.
6. Hubert De Fraysseix, János Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990. doi:10.1007/bf02122694.
7. István Fáry. On straight line representation of planar graphs. *Acta Sci. Math.(Szeged)*, 11:229–233, 1948.
8. Jim Geelen, Anjie Guo, and David McKinnon. Straight line embeddings of cubic planar graphs with integer edge lengths. *J. Graph Theory*, 58(3):270–274, 2008. doi:10.1002/jgt.20304.
9. Heiko Harborth, Arnfried Kemnitz, Meinhard Möller, and Andreas Süssenbach. Ganzzahlige planare Darstellungen der platonischen Körper. *Elem. Math.*, 45(5):118–122, 1987.
10. Arnfried Kemnitz and Heiko Harborth. Plane integral drawings of planar graphs. *Discrete Mathematics*, 236(1-3):191–195, 2001. doi:10.1016/s0012-365x(00)00442-8.
11. Michael Kleber. Encounter at far point. *The Mathematical Intelligencer*, 30:50–53, 2008. doi:10.1007/bf02985756.
12. Ronald C. Read. *A new method for drawing a planar graph given the cyclic order of the edges at each vertex*. Faculty of Mathematics, University of Waterloo, 1986.
13. Walter Schnyder. Embedding planar graphs on the grid. In *Proc. SODA 1990*, 1990. URL: https://dl.acm.org/doi/10.5555/320176.320191.
14. Sherman K Stein. Convex maps. *Proc. Am. Math. Soc.*, 2(3):464–466, 1951.
15. Timothy Sun. Rigidity-theoretic constructions of integral fary embeddings. In *Proc. CCCG 2011*, 2011. URL: http://www.cccg.ca/proceedings/2011/papers/paper20.pdf.
16. William Thomas Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 3(1):743–767, 1963. doi:10.1112/plms/s3-13.1.743.
17. Klaus Wagner. Bemerkungen zum Vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 46:26–32, 1936.

A Poster

Drawing Trees and Cacti with Integer Edge Lengths on a Polynomial-Size Grid

Henry Förster, Stephen Kobourov, Jacob Miller, and Johannes Zink

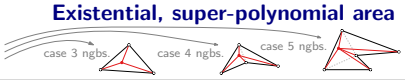
The history of planar straight-line graph drawings

Conjecture — Szele conjectures in a discussion about Kuratowski's characterization of planar graphs that every planar graph can be drawn with straight-line segments. — 1940s

1936, 1948, 1951 — Wagner, Fáry, and Stein prove independently that every planar graph has a crossings-free drawing with edges drawn by straight-line segments. Fáry refers to Szele's conjecture.

Existential, super-polynomial area

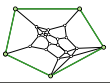
triangulate graph; inductively remove inner vertex with max. degree 5; re-insertion in three cases:



Efficient algorithm, super-polynomial area



1. fix outer face
2. place in barrycenter of neighbors by solving system of linear equations



Tutte, later Chiba, Yamanouchi, and Nishizeki, and then Read present efficient algorithms for generating planar straight-line drawings. — 1963, 1984, 1986

1990 — The algorithm of de Fraysseix, Pach and Polack, which relies on canonical orders, and Schnyder's algorithm, which uses a decomposition of the graph into three trees, provide planar straight-line drawings on the $\mathcal{O}(n) \times \mathcal{O}(n)$ grid in linear time.

Efficient algorithm, polynomial-size grid

1. find cannonical order
2. incrementally draw with shift method



1. decomposition into 3 trees
2. place vtcs. w.r.t. to relative location to the 3 roots



Conjecture — Harborth, Kemnitz, Möller, and Süssenbach conjecture that every planar graph admits a planar straight-line drawing with all edge lengths being integers. Later, Kleber strengthens the conjecture by requiring that the vertices lie on integer grid points. — 1987, 2001, 2008

2011, 2024 — Sun uses rigidity matrices to prove that all planar (2, 3)-sparse graphs admit a planar straight-line drawing with integer edge lengths. A graph G is (2, 3)-sparse if every subgraph G' of G has at most $2|V(G')| - 3$ edges. These graphs include series-parallel, outerplanar, and planar bipartite graphs. Later, Chang and Sun show this result also for all planar 4-regular graphs.

Existential, integer length, no integer grid

rigidity matrix of the drawing of K_3 below: $2 \cdot \begin{pmatrix} 1 & -2 & -1 & 2 & 0 & 0 \\ 3 & -1 & 0 & 0 & -3 & 1 \\ 0 & 0 & 2 & 1 & -2 & -1 \end{pmatrix}$





Existential, integer length, super-polynomial grid

Berry's theorem:
$a, b, c \in \mathbb{R}^2$ non-collinear, $d(a,b)$, $d(a,c)^2$, $d(b,c)^2$ rational
$\Rightarrow$ set of points being a rational distance from each of a, b, and c form a dense subset



Start with a planar s.-l. drawing. Due to Berry's thm., $\exists$ points to slightly perterub vertices and get integer edge lengths.



Geelen, Guo, and McKinnon use Berry's theorem to show that every planar max-degree-3 graph admits a planar straight-line drawing where all vertices lie on grid points and all edges have integer lengths. Biedl observes that this technique can be applied to all planar (2, 1)-sparse graphs, which include all max-degree-4 graphs that are non-4-regular. Independently, Benediktovich shows this result for max-degree-4 graphs that are non-4-regular and for planar 3-trees. — 2008, 2011, 2013

2011 — Biedl shows how to draw planar max-degree-3 graphs with integer edge lengths and integer coordinates. She employs an algorithm by Kant from 1992 for drawing 3-connected 3-regular graphs on the hexagonal grid.

Efficient algorithm, integer length, super-polynomial grid

Use Kant's algorithm to draw 3-connected 3-regular graphs on a hexagonal grid.

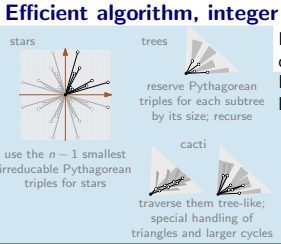


Skew the drawing s.t. the three resulting slopes are part of Pythagorean triples.



Draw non-3-connected parts independently, scale and glue

Efficient algorithm, integer length, polynomial-size grid



For the case of 3-connected 3-regular planar graphs, Biedl's algorithm yields drawings on the $\mathcal{O}(n) \times \mathcal{O}(n)$ grid. We show, by inspecting primitive Pythagorean triples, that a planar straight-line drawing with integer edge lengths can be obtained in polynomial time on the integer grid of size ...

- $\mathcal{O}(n) \times \mathcal{O}(n)$ for binary trees and stars (optimal for stars),
- $\mathcal{O}(\ell d) \times \mathcal{O}(\ell d) \subseteq \mathcal{O}(n^2) \times \mathcal{O}(n^2)$ for trees with ℓ leaves and depth d, and
- $\mathcal{O}(t(o^2+\ell^2)+\ell d + o(\ell+d+o)) \times \mathcal{O}(t(o^2+\ell^2)+\ell d + o(\ell+d+o)) \subseteq \mathcal{O}(n^3) \times \mathcal{O}(n^3)$ for cacti with t triangles, o cycles, ℓ leaves and diameter d.

our contribution

2011, 2025

Full version: https://arxiv.org/abs/2509.04168

Future

- Find lower bounds on the grid size for general trees and cacti. Reduce the grid size if possible.
- Do polynomial-size grids also suffice for larger graph classes (e.g., outerplanar graphs, max-deg-3 graphs, 4-regular graphs, k-trees)?
- Is Harborth et al.'s conjecture true? Is Kleber's conjecture true?

$n :=$ number of vertices in a graph

Defective Linear Layouts of Graphs

Michael A. Bekos
University of Ioannina, Greece

Carla Binucci
University of Perugia, Italy

Emilio Di Giacomo
University of Perugia, Italy

Walter Didimo
University of Perugia, Italy

Luca Grilli
University of Perugia, Italy

Maria Eleni Pavlidi
University of Ioannina, Greece

Alessandra Tappini
University of Perugia, Italy

Alexandra Weinberger
FernUniversität in Hagen, Germany

Abstract

A linear layout of a graph defines a total order of the vertices and partitions the edges into either *stacks* or *queues*, i.e., crossing-free and non-nested sets of edges along the order, respectively. In this work, we study defective linear layouts that allow forbidden patterns among edges of the same set. Our focus is on *k-defective stack layouts* and *k-defective queue layouts*, in which the *conflict graph* representing the forbidden patterns among the edges of each stack or queue has maximum degree at most k.

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

Keywords and phrases Linear layouts, stack layouts, queue layouts, defective layouts

Digital Object Identifier 10.4230/LIPIcs.GD.2025.49

Category Poster Abstract

Funding *Bekos* and *Pavlidi* were supported by HFRI Grant No: 26320.

Acknowledgements Research started at the Bertinoro Workshop on Graph Drawing 2024.

1 Introduction

Stack [3] and queue [5] layouts are key concepts in topological graph theory, where vertices are ordered and edges are partitioned into non-crossing (stacks) or non-nested (queues) sets. The goal is to minimize the number of stacks/queues – known as the *stack/queue number*. We introduce k-defective linear layouts, a generalization of stack/queue layouts that allows some forbidden patterns (edge crossings or nestings), controlled by a conflict graph with maximum degree k. Classical layouts correspond to the case $k = 0$. Our contributions include: (i) characterizations of k-defective layouts; (ii) bounds on edge density; and (iii) bounds or exact values of k-defective numbers for specific graph families.

2 Queue Layouts

We first bound the density of graphs admitting k-defective h-queue layouts. Let $v_0 \prec \cdots \prec v_{n-1}$ be a vertex ordering of a graph G. Partition the edges into $n-1$ *classes*, where edges (v_{i_1}, v_{j_1}) and (v_{i_2}, v_{j_2}) belong to the same class if $\lfloor \frac{i_1+j_1}{2} \rfloor = \lfloor \frac{i_2+j_2}{2} \rfloor$. We denote by C_i the class whose maximum number of edges is i $(i = 1, \ldots, n-1)$. Note that if C_i has i edges, it induces a path.

▶ **Lemma 1.** *Let $\mathcal{L}$ be a k-defective h-queue layout of a graph G. Every defective queue of $\mathcal{L}$ contains at most $k+2$ edges of each class.*



© Michael A. Bekos, Carla Binucci, Emilio Di Giacomo, Walter Didimo, Luca Grilli, Maria Eleni Pavlidi, Alessandra Tappini, and Alexandra Weinberger;
licensed under Creative Commons License CC-BY 4.0

▶ **Theorem 2.** *Any n-vertex graph that admits a k-defective h-queue layout has at most $h(k+2)\left(n - \frac{h(k+2)+1}{2}\right)$ edges, for $n \geq h(k+2)+1$.*

Sketch. Let $\mathcal{L}$ be a k-defective h-queue layout of a graph G with n vertices and m edges. Classes with at most $h(k+2)$ edges are *small*, otherwise are *large*. Let n_l be the number of large classes, and m_s the number of edges in small classes. We claim $m \leq n_l h(k+2) + m_s$. Suppose, for contradiction, that $m > n_l h(k+2) + m_s$. Then some large class must have more than $h(k+2)$ edges, implying a queue with more than $k+2$ edges from that class, contradicting Lemma 1. Since each class has at most i edges, we have $n_l = n - 1 - h(k+2)$, and $m_s \leq \sum_{i=1}^{h(k+2)} i = \frac{h(k+2)(h(k+2)+1)}{2}$. Therefore, $m \leq (n-1-h(k+2))h(k+2) + \frac{h(k+2)(h(k+2)+1)}{2} = h(k+2)\left(n - \frac{h(k+2)+1}{2}\right)$. Finally, $h(k+2)\left(n - \frac{h(k+2)+1}{2}\right) \leq \frac{n(n-1)}{2}$ for $n \geq h(k+2)+1$. ◀

We next consider defectiveness and queue number. In this content, outerplanar graphs have queue number 2 [7], but not all admit 1-queue layouts with bounded defectiveness. Outer 1-planar graphs have queue number at most 42 [2]; we show that their 1-defective queue number is 2, so their queue number is at most 3.
For general planar graphs, we can prove an upper bound of 33 on their 1-defective queue number by adapting a well-known technique by Dujmovic et al. [4].
We next establish bounds on the k-defective queue number of K_n that are tight for $k = 1$.

▶ **Theorem 3.** *The k-defective queue number of K_n is at least $\left\lceil \frac{n-1}{k+2} \right\rceil$ and at most $\lceil \frac{n-1}{l} \rceil$, where $l = \left\lfloor \frac{3+\sqrt{8k+1}}{2} \right\rfloor$.*

Sketch. For the lower bound, consider a k-defective queue layout $\mathcal{L}$ of K_n. The edges can be partitioned into classes $C_1, \ldots, C_{n-1}$. Since C_{n-1} has $n-1$ edges and at most $k+2$ edges can share a defective queue (Lemma 1), at least $\left\lceil \frac{n-1}{k+2} \right\rceil$ queues are needed. We prove the upper bound via an explicit construction. For h to be specified, assign to each defective queue q_a (for $0 \leq a \leq h$) all edges with hop-size $al + i$ for $1 \leq i \leq l$. The edges with the most nestings are the $(al+l)$-hop edges, each nesting $\frac{1}{2}(l-2)(l-1)$ edges, ensuring the layout is k-defective. To cover all $\frac{n(n-1)}{2}$ edges of K_n, note that each q_a contains $\sum_{i=1}^{l}(n-(al+i))$ edges. For $h = \lceil \frac{n-1}{l} \rceil$, the total edges assigned satisfy $\sum_{a=0}^{h-1}\sum_{i=1}^{l}(n-(al+i)) \geq \frac{n(n-1)}{2}$. ◀

We give bounds on the k-defective queue number of $K_{n,n}$, considering both the general case and the *separated setting*, where one part precedes the other [1].

▶ **Corollary 4.** *The 1-defective queue number of $K_{n,n}$ in the separated setting is $\lceil \frac{2n-1}{3} \rceil$. For $k > 1$, it ranges between $\left\lceil \frac{2n-1}{k+2} \right\rceil$ and $\lceil \frac{2n-1}{l} \rceil$ in the separated setting, while in the non-separated setting it ranges between $\left\lceil \frac{n-1}{k+2} \right\rceil$ and $\lceil \frac{2n-1}{l} \rceil$, where $l = \left\lfloor \frac{3+\sqrt{8k+1}}{2} \right\rfloor$.*

3 Stack Layouts

A graph admits a k-defective h-stack layout if and only if its edges can be partitioned into h defective stacks, each forming an outer k-planar subgraph. This yields the following characterization.

▶ **Theorem 5.** *A graph has k-defective stack number 1 if and only if it is outer k-planar.*

We leverage the characterization given above to obtain bounds on the edge density of the graphs admitting k-defective h-stack layouts.

▶ **Theorem 6** (Ábrego et al. [8], Pach et al. [6]). *Any n-vertex graph that admits a k-defective 1-stack layout has at most:* **(i)** $\frac{5}{2}n-4$ *edges, if* $k=1$, **(ii)** $3n-5$ *edges, if* $k=2$, **(iii)** $\frac{13}{4}n-\frac{11}{2}$ *edges, if* $k=3$, **(iv)** $O(\sqrt{k}n)$, *otherwise. Also, the first three bounds are tight.*

▶ **Theorem 7.** *An n-vertex graph with a 1-defective h-stack layout has at most* $\left(\frac{3}{2}h+1\right)n-4h$ *edges.*

In the following, we present bounds on the k-defective stack numbers of K_n and $K_{n,n}$. For $k=1$, our upper bounds are tight (up to small constants).

▶ **Theorem 8.** *The k-defective stack number of K_n is at least* $\left\lceil \frac{n}{2k+2} \right\rceil$ *and at most* $\left\lceil \frac{n}{l+2} \right\rceil$, *where* $l=\left\lfloor \frac{-1+\sqrt{8k+1}}{2} \right\rfloor$.

▶ **Corollary 9.** *The 1-defective stack number of $K_{n,n}$ in the separated setting is at least* $\lceil \frac{n}{2} \rceil$ *and at most* $\lceil \frac{2n}{3} \rceil$, *while in the non-seperated setting it is at least* $\lceil \frac{n}{4} \rceil$ *and at most* $\lceil \frac{n}{2} \rceil$.

▶ **Theorem 10.** *The k-defective stack number of $K_{n,n}$ in the separated setting is at least* $\left\lceil \frac{n}{2k+2} \right\rceil$ *and at most* $\lceil \frac{n}{l} \rceil$, *where* $l=\sqrt{k}+1$.

4 Open Problems

Our work raises several new open problems, which we list below.

- Extend to other linear layouts (e.g., deques, riques) and mixed settings.
- Explore defects beyond bounded conflict graph degree, such as bounding diameter.
- Research questions from our results: (i) complexity of recognizing graphs with k-defective h-queue layouts for $k,h\geq 1$, (ii) improve bounds on k-defective stack and queue numbers of K_n for $k>1$, and (iii) study other graph classes.

References

1 Jawaherul Md. Alam, Michael A. Bekos, Martin Gronemann, Michael Kaufmann, and Sergey Pupyrev. The mixed page number of graphs. *Theor. Comput. Sci.*, 931:131–141, 2022. doi:10.1016/J.TCS.2022.07.036.

2 Michael A. Bekos, Martin Gronemann, and Chrysanthi N. Raftopoulou. An improved upper bound on the queue number of planar graphs. *Algorithmica*, 85(2):544–562, 2023. doi:10.1007/s00453-022-01037-4.

3 Frank Bernhart and Paul C. Kainen. The book thickness of a graph. *J. Comb. Theory, Ser. B*, 27(3):320–331, 1979. doi:10.1016/0095-8956(79)90021-2.

4 Vida Dujmović, Gwenaël Joret, Piotr Micek, Pat Morin, Torsten Ueckerdt, and David R. Wood. Planar graphs have bounded queue-number. *J. ACM*, 67(4):22:1–22:38, 2020. doi:10.1145/3385731.

5 Lenwood S. Heath and Arnold L. Rosenberg. Laying out graphs using queues. *SIAM J. Comput.*, 21(5):927–958, 1992. doi:10.1137/0221055.

6 János Pach and Géza Tóth. Graphs drawn with few crossings per edge. *Comb.*, 17(3):427–439, 1997. doi:10.1007/BF01215922.

7 S. Rengarajan and C. E. Veni Madhavan. Stack and queue number of 2-trees. In Ding-Zhu Du and Ming Li, editors, *Computing and Combinatorics, First Annual International Conference, COCOON '95, Xi'an, China, August 24-26, 1995, Proceedings*, volume 959 of *Lecture Notes in Computer Science*, pages 203–212, 1995. doi:10.1007/BFB0030834.

8 Bernardo M. Ábrego, Joseph Dandurand, Silvia Fernández-Merchant, Emilie Lagoda, and Yury Sapozhnikov. Book crossing numbers of the complete graph and small local convex crossing numbers. *CoRR*, abs/1607.00131, 2024. URL: https://arxiv.org/abs/1607.00131.

gd 25

Defective Linear Layouts

Michael A. Bekos [1] Carla Binucci [2] Emilio Di Giacomo [2] Walter Didimo [2] Luca Grilli [2]
Maria Eleni Pavlidi [1] Alessandra Tappini [2] Alexandra Weinberger [3]

1. University of Ioannina 2. 3. FernUniversität in Hagen

Linear Layouts of Graphs

- Given a graph, order $\prec$ its vertices
- and partition its edges into pages s.t. each page has a property. ⊢ stacks ⊢ queues

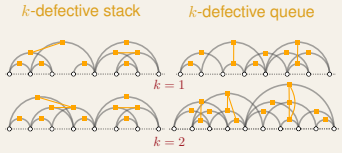


In a stack the edges are crossing-free. In a queue the edges are non-nested.

Stack/queue number = minimum required number of stacks/queues.

Conflict Graph

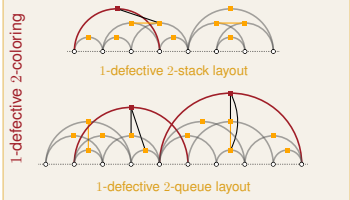
- Nodes → edges in the same page.
- Edges → forbidden patterns.
- Degree at most k at every page → k-defects.



k-defective stack/queue number = minimum required number of k-defective stacks/queues.

Defective layouts and colorings

$\exists$ k-defective layout with h stacks (queues) iff $\exists \prec$ s.t. the conflict graph admits an h-coloring with monochromatic components of degree $\leq k$

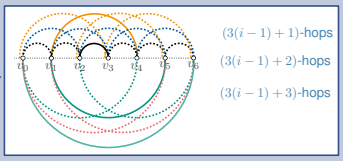


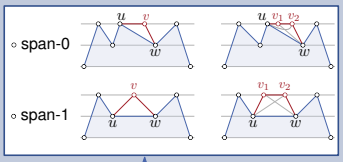
Results

k-defective queue layouts

k	h	Edge density bounds	Tight
>1	>1	$h(k+2)\left(n-\frac{h(k+2)+1}{2}\right)$	
1	>1	$3h\left(n-\frac{3h+1}{2}\right)$	✓
2	1	$\frac{10}{3}n-\frac{21+r}{3}, r=\text{mod}3$	✓
1	1	$3n-6$	✓

$h = \#$queues

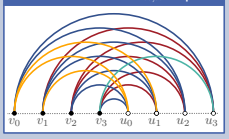




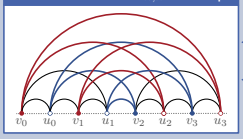
Upper Bound for $K_{n,n}$

Directly from the queue number bounds of K_{2n}.

Lower bound for $K_{n,n}$ sep.

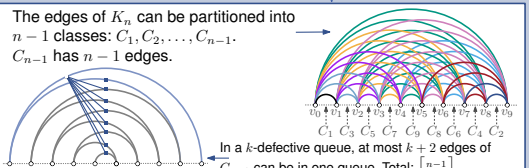


Lower bound for $K_{n,n}$ non-sep.



k-defective q.n.	Lower Bound	Upper Bound
Outer 1-planar ($k=1$)	2	2
Planar ($k=1$)	2	33
K_n	$\lceil\frac{n-1}{k+2}\rceil$	$\lceil\frac{n-1}{l_1}\rceil$ $l_1=\lfloor\frac{3+\sqrt{8k+1}}{2}\rfloor$
$K_{n,n}$ (sep)	$\lceil\frac{2n-1}{k+2}\rceil$	$\lceil\frac{2n-1}{l_1}\rceil$
$K_{n,n}$ (non sep)	$\lceil\frac{n-1}{k+2}\rceil$	$\lceil\frac{2n-1}{l_1}\rceil$

The edges of K_n can be partitioned into $n-1$ classes: $C_1, C_2, \ldots, C_{n-1}$. C_{n-1} has $n-1$ edges.



In a k-defective queue, at most $k+2$ edges of C_{n-1} can be in one queue. Total: $\lceil\frac{n-1}{k+2}\rceil$

k-defective stack layouts

Theorem ([1, 2])

Any n-vertex graph that admits a k-defective h-stack layout has at most

1. $(\frac{5}{2}h+1)n-4h$ edges, if $k=1$
2. $(3h+1)n-5h$ edges, if $k=2$
3. $(\frac{13}{4}h+1)n-\frac{11}{2}h$, if $k=3$
4. $O(\sqrt{k}hn)$, otherwise

k	h	Edge density bounds	Tight
>1	>1	$O(\sqrt{k}hn)$	
1	>1	$(\frac{3}{2}h+1)n-4h$	
2	1	$3n-5$	✓
1	1	$\frac{5}{2}n-4$ [3, 4]	✓

$h = \#$stacks

Proof

Outer 1-planar graphs have $\leq \frac{5n}{2}-4$ edges ⇒

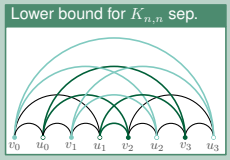
The 1st page has so many edges, while the rest have at most $\frac{3n}{2}-4$ (as the boundary edges are in the 1st page).

Total edges: $\frac{5n}{2}-4+(h-1)\cdot(\frac{3n}{2}-4)=(\frac{3n}{2}h+1)n-4h$

Theorem

A graph has k-defective stack number 1 iff it is outer k-planar.

Lower bound for $K_{n,n}$ sep.



Lower bound for K_n

In any vertex order of K_n, there exist $\frac{n}{2}$ pairwise crossing edges.

Thus, at most $k+1$ of them lie in the same defective stack.

k-defective s.n.	Lower Bound	Upper Bound
Outer k-planar	1	1
Planar ($k=1$)	2	4 [5]
K_n	$\lceil\frac{n}{2k+2}\rceil$ [6]	$\lceil\frac{n}{l_2+2}\rceil$ $l_2=\lfloor\frac{-1+\sqrt{8k+1}}{2}\rfloor$
$K_{n,n}$ (sep)	$\lceil\frac{n}{k+1}\rceil$	$\lceil\frac{2n}{l_2+2}\rceil$
$K_{n,n}$ (non sep)	$\lceil\frac{n}{2k+2}\rceil$	$\lceil\frac{n}{l_3}\rceil$ $l_3=\sqrt{k}+1$

Problems

- Extend to other linear layouts (e.g., deques, riques) and mixed settings.
- Explore new defect types, such as bounded diameter in conflict graphs.
- Study the complexity of recognizing graphs that admit k-defective h-queue layouts.

References

1. J. Pach, G. Tóth. Graphs drawn with few crossings per edge. Comb., 17(3):427–439,578 1997.
2. B. M. Ábrego, J.h Dandurand, S. Fernández-Merchant, E. Lagoda, Y. Sapozhnikov. Book crossing numbers of the complete graph and small local convex crossing numbers. CoRR, abs/1607.00131, 2024.
3. C. Auer, C. Bachmaier, F. J. Brandenburg, A. Gleißner, K. Hanauer, D. Neuwirth, J. Reislhuber. Outer 1-planar graphs. Algorithmica, 74(4):1293–1320, 2016.
4. W. Didimo. Density of straight-line 1-planar graph drawings. Inf. Process. Lett., 113(7):236– 240, 2013.
5. M. Yannakakis. Planar graphs that need four pages. J. Comb. Theory, Ser. B, 145:241–263, 2020.
6. F. Bernhart, P. C. Kainen. The book thickness of a graph. J. Comb. Theory, Ser. B, 27(3):320–331, 1979.

Reeb Lobsters Are 1-Planar

Maarten Löffler
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Miriam Münch
Faculty of Computer Science and Mathematics, University of Passau, Germany

Ignaz Rutter
Faculty of Computer Science and Mathematics, University of Passau, Germany

Abstract

Very recently, Chambers, Fasy, Hosseini Sereshgi and Löffler [3] showed that every Reeb caterpillar admits a crossing-free drawing. It turns out that this does not hold for Reeb lobsters but we show that these graphs admit drawings with at most one crossing per edge.

2012 ACM Subject Classification Mathematics of computing → Graphs and surfaces; Mathematics of computing → Topology

Keywords and phrases Reeb graphs, layered drawings, local crossing number

Digital Object Identifier 10.4230/LIPIcs.GD.2025.50

Category Poster Abstract

Funding Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 541433306.

1 Introduction

A *layered* graph (G, h) is a graph together with pairwise-distinct preassigned heights of its vertices. In a drawing of a layered graph each vertex v is mapped to a unique point $(x, h(v)) \in \mathbb{R}^2$ and every edge is mapped to a continuous y-monotone curve [1, 7]. For a vertex v in a layered graph we denote the horizontal line through v by ℓ_v. Let $\deg_\downarrow(v)$ denote the number of neighbors w of v with $h(w) < h(v)$. Similarly we let $\deg_\uparrow(v)$ denote the number of neihgbors w of v with $h(w) > h(v)$. A *Reeb graph* (G, h), introduced in [6], is a layered graph such that for every vertex v we have $\deg(v) \in \{1, 3\}$, $\deg_\downarrow(v) \leq 2$ and $\deg_\uparrow(v) \leq 2$. Reeb graphs can be computed effieciently [4, 5] and capture the evolution of level sets of functions from a topological space to the real numbers. They have many applications in topological data analysis and visualization [8, 2].

A *caterpillar* is a tree C such that removing all leaves from C yields a path, called the *spine* of C. A tree L such that removing all leaves yields a caterpillar is a *lobster*. The spine of a lobster L is the spine of the caterpillar we obtain by removing all leaves from L. The *local crossing number* $\mathrm{lcr}(G)$ of a (layered) graph G is the smallest integer k such that G admits a drawing with at most k crossings per edge. A graph G is *k-planar* if $\mathrm{lcr}(G) \leq k$.

2 Our Results

First we show that crossings are needed to draw Reeb lobsters.

▶ **Theorem 1.** *Not every Reeb lobster admits a crossing-free drawing.*

Proof. Consider the lobster G shown in Figure 1(a). Assume that G admits a crossing-free drawing Γ. Let G' be the graph we obtain from G by adding two new independent vertices x, y with $h(x) > h(v)$ and $h(y) < h(v)$ for every $v \in V$ such that x is adjacent exactly to the

© Maarten Löffler, Miriam Münch, and Ignaz Rutter;
licensed under Creative Commons License CC-BY 4.0

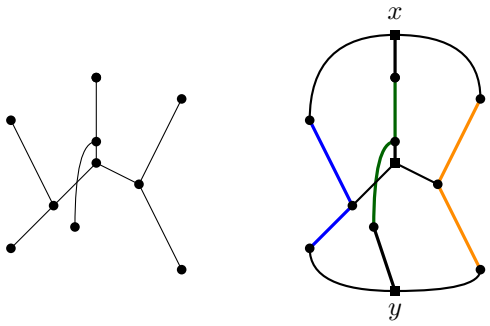


Figure 1 (a) A Reeb lobster and (b) its extension to a $K_{3,3}$-subdivision.

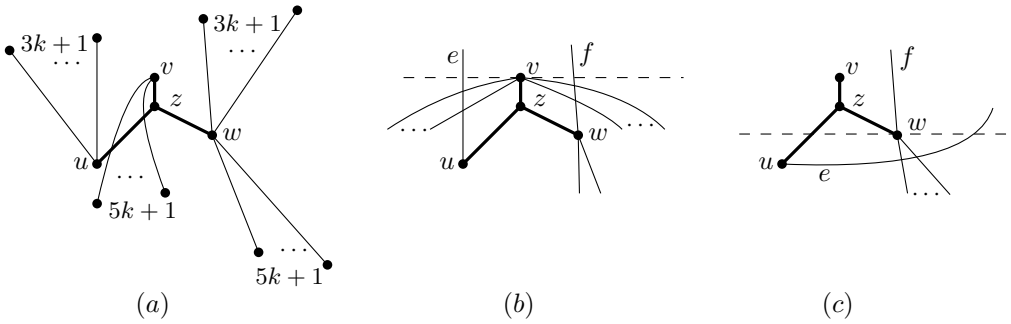


Figure 2 Illustration of the proof of Theorem 2.

three vertices with greatest heights and y is adjacent exactly to the three vertices with lowest heights; see Figure 1(b). Then it is easy to extend Γ to a crossing-free drawing of G' but G' is a $K_{3,3}$-subdivision and thus non-planar; a contradiction. ◀

Next we show that the local crossing-number of a layered lobster may be arbitrarily large.

▶ **Theorem 2.** *For every $k \in \mathbb{N}$ there is a layered lobster L such that* $\mathrm{lcr}(L) > k$.

Proof. For a fixed $k \in \mathbb{N}$ consider the lobster L sketched in Figure 2(a). We call a set F of l edges an *l-fan* if all edges in F share a common endpoint c, the remaining endpoints have degree 1 and the heights of these degree-1 endpoints lie either all above $h(c)$ or all below $h(c)$. For each of the four fans in L the heights of the corresponding degree-1 vertices are chosen such that each other vertex either lies below all of them or above all of them. Assume for the sake of contradiction that there exists a drawing Γ of L with at most k crossings per edge. Let u' be a degree-1 vertex adjacent to u, whose incident edge does not cross an edge incident to z; such a vertex exists as there are $3k+1$ degree-1 vertices adjacent to u and the edges incident to z have at most $3k$ crossings. Analogously, there exists a degree-1 vertex w' adjacent to w with $h(w') > h(w)$, whose incident edge does not cross an edge incident to z. Similarly, we can find sets V'' and W'' each consisting of $2k+1$ degree-1 vertices adjacent to v and w, respectively, that lie below w and whose incident edges do not cross an edge incident to z. Let Γ' denote the drawing induced by $\{u, u', v, w, w', z\} \cup V'' \cup W''$, where we possibly reroute zu and zw so that they do not cross each other. Note that all edges incident to z are crossing-free in Γ'. Consider the edges $e = uu'$ and $f = ww'$. Without loss of generality assume that zu crosses ℓ_w left of w. Since the path uzw is crossing-free, f crosses ℓ_z to the right of z and since zv is crossing-free, f crosses ℓ_v to the right of v. We distinguish two cases based on whether e crosses ℓ_v on the left or on the right side of v.

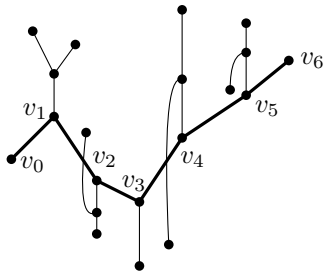


Figure 3 Illustration of the construction of a 1-planar drawing of a lobster that is a Reeb graph.

Assume that e crosses ℓ_v on the left side of v; see Figure 2(b). As v lies between e and f on ℓ_v and the edges incident to z are crossing-free in Γ', each edge connecting v to V'' crosses e or f. Since $|V''| = 2k + 1$, e or f has more than k crossings; a contradiction.

It remains to consider the case where e crosses ℓ_v on the right side of v; see Figure 2(c). Note that in this case e crosses the horizontal line ℓ_w to the right of w since the path wzv is uncrossed. Since the path zuu' starts and ends above w, crosses ℓ_w once to the left of w and once to the right of w, and zu is crossing-free, all edges connecting w to W'' cross e. Since $|W''| = 2k + 1$ it follows that e has more than k crossings; a contradiction. ◀

Finally we show that Reeb lobsters admit a drawing with at most one crossing per edge.

▶ **Theorem 3.** *Every Reeb lobster is 1-planar.*

Proof. Let L be a Reeb graph. We consider an *extended spine* of L; i.e, a path between degree-1 vertices of L that contains the spine. Let $\{v_0, \dots, v_k\}$ be an extended spine of L such that for every $0 < i < k$ the two neighbors of v_i are v_{i-1} and v_{i+1}. We start by drawing the extended spine x-monotone; see Figure 3 for an illustration. Next we insert all vertices of L that are adjacent to the extended spine crossing-free by assigning each such vertex u an x-coordinate close to the x-coordinate of its neighbor v on the extended spine. It remains to add the missing leaves of L that have a neighbor that is not adjacent to a vertex on the extended spine. Let w be such a leaf of L with neighbor u. Let v_i be the neighbor of u on the extended spine. If $h(v_i)$ does not lie between $h(w)$ and $h(u)$, then we can easily insert w crossing-free by assigning it an x-coordinate close to or equal to the x-coordinate of u. This does not result in a crossing. Hence assume that $h(w) < h(v_i) < h(u)$ or $h(w) > h(v_i) > h(u)$. Now we assign w an x-coordinate between the x-coordinates of v_i and v_{i-1}. Then wu crosses the edge v_iv_{i-1}. Hence by construction for every such leaf w, the edge wu receives at most one crossing. Recall that $\deg_{\downarrow}(u) \leq 2$, $\deg_{\uparrow}(u) \leq 2$. Thus every u that is neither a leaf of L nor part of the extended spine has at most one neighbor w with $h(w) < h(v_i) < h(u)$ or $h(w) > h(v_i) > h(u)$. Hence for every $0 < i < k$ also v_iv_{i-1} receives at most one crossing. Thus the construction described above yields a 1-planar drawing of L. ◀

3 Conclusion

Our results show that the property to be a Reeb graph yields an upper bound of 1 for the otherwise unbounded local crossing number of layered lobsters. An interesting question is how this generalizes to more general Reeb trees.

References

1 Oliver Bastert and Christian Matuszewski. *Layered Drawings of Digraphs*, pages 87–120. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. doi:10.1007/3-540-44969-8_5.

2 Silvia Biasotti, Daniela Giorgi, Michela Spagnuolo, and Bianca Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical computer science*, 392(1-3):5–22, 2008. doi:10.1016/j.tcs.2007.10.018.

3 Erin W. Chambers, Brittany Terese Fasy, Erfan Hosseini Sereshgi, and Maarten Löffler. Drawing reeb graphs. In *Combinatorial Algorithms - 36th International Workshop, IWOCA 2025*, volume 15885 of *Lecture Notes in Computer Science*, pages 58–71. Springer, 2025. doi:10.1007/978-3-031-98740-3_5.

4 William Harvey, Yusu Wang, and Rephael Wenger. A randomized $O(m \log m)$ time algorithm for computing Reeb graphs of arbitrary simplicial complexes. In *Proceedings of the 2010 annual symposium on Computational geometry*, SoCG '10, pages 267–276. ACM, 2010. doi:10.1145/1810959.1811005.

5 Salman Parsa. A deterministic O (m log m) time algorithm for the reeb graph. *Discret. Comput. Geom.*, 49(4):864–878, 2013. doi:10.1007/s00454-013-9511-3.

6 Georges Reeb. Sur les points singuliers d'une forme de pfaff completement integrable ou d'une fonction numerique [on the singular points of a completely integrable pfaff form or of a numerical function]. *Comptes Rendus Acad. Sciences Paris*, 222:847–849, 1946. URL: https://cir.nii.ac.jp/crid/1571417125676878592.

7 Kozo Sugiyama, Shôjirô Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics, SMC*, 11(2):109–125, 1981. doi:10.1109/TSMC.1981.4308636.

8 Lin Yan, Talha Bin Masood, Raghavendra Sridharamurthy, Farhan Rasheed, Vijay Natarajan, Ingrid Hotz, and Bei Wang. Scalar field comparison with topological descriptors: Properties and applications for scientific visualization. In *Computer Graphics Forum*, volume 40, pages 599–633. Wiley Online Library, 2021. doi:10.1111/cgf.14331.

Reeb Lobsters are 1-Planar

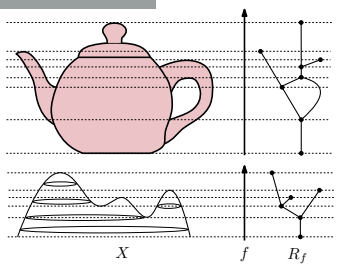
Maarten Löffler[1] • Miriam Münch[2] • Ignaz Rutter[2]

Utrecht University

UNIVERSITÄT PASSAU

Reeb Graphs

- Given: topological space X, function $f : X \to \mathbb{R}$
- for two points $x, y \in X$ we have $x \sim y$ iff
  - $f(x) = f(y) = a$ and
  - x, y are in the same path-connected component of $f^{-1}(a)$
- Reeb graph R_f is the quotient space $X/\sim$



- capture evolution of level sets of functions from a topological space to the real numbers
- applications in topological data analysis and visualization (e.g. [2, 3])

Properties of Reeb Graphs

Layered graph with height function $h : V \to \mathbb{R}$ s.t. for each $v \in V$

- $h(v) \neq h(w)$ for any $w \neq v$
- $\deg(v) \in \{1, 3\}$
- downward degree $\deg_\downarrow(v) \leq 2$, upward degree $\deg_\uparrow(v) \leq 2$

Question: How to draw Reeb graphs with few crossings?

Known: Reeb caterpillars can be drawn crossing-free (Chambers, Fasy, Sereshgi, Löffler '25 [1])



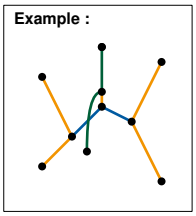
What about lobsters?



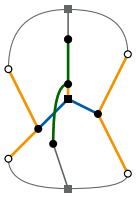
Theorem 1

Not every Reeb lobster admits a planar drawing.

Example :



add 2 vertices crossing-free

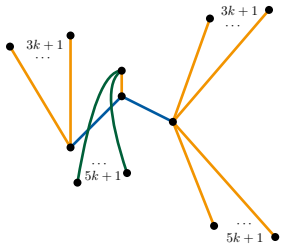


$K_{3,3}$-subdivision

no planar drawing ⇐ no planar drawing

Theorem 2

For every $k \in \mathbb{N}$ there is a layered lobster that is not k-planar.



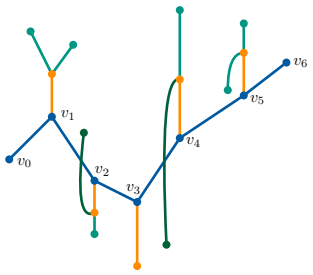
Theorem 3

Every Reeb lobster is 1-planar.

Construction :

- Draw **extended spine** x-monotone (path between degree-1 vertices that contains the spine)
- Add **caterpillar leaves** crossing-free
- Add **lobster leaves** as follows:
  Let w be a lobster leaf with neighbor u and let v_i be the spine-neighbor of u.
  - **Case 1:** $h(v_i) = \max(h(u), h(w), h(v_i))$ or $h(v_i) = \min(h(u), h(w), h(v_i))$
    → add w crossing-free
  - **Case 2:** $h(u) < h(v_i) < h(u)$ or $h(u) < h(v_i) < h(w)$
    → add w s.t. wu crosses $v_{i-1}v_i$

  By construction: wu crossed at most once
  $\deg_\downarrow(u) \leq 2$, $\deg_\uparrow(u) \leq 2 \Rightarrow v_{i-1}v_i$ crossed at most once



Conclusion

First step towards efficient crossing minimization for Reeb graphs → How do our results generalize to more general Reeb trees?

1. E. W. Chambers, B. T. Fasy, E. Hosseini Sereshgi, and M. Löffler. Drawing Reeb Graphs. *CoRR*, abs/2504.21329, 2025.
2. S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical computer science*, 392(1-3):5-22, 2008.
3. L. Yan, T. B. Masood, R. Sridharamurthy, F. Rasheed, V. Natarajan, I. Hotz, and B. Wang. Scalar field comparison with topological descriptors: Properties and applications for scientific visualization. *Computer Graphics Forum* , 40(3):599-633, 2021.

Graph Tiles

Oswin Aichholzer
TU Graz, Austria

Robert Ganian
TU Wien, Austria

Phillip Keldenich
TU Braunschweig, Germany

Maarten Löffler
Utrecht University, The Netherlands

Gert Meijer
Stenden University of Applied Sciences, Leeuwarden, The Netherlands

Alexandra Weinberger
FernUniversität in Hagen, Germany

Carola Wenk
Tulane University, New Orleans, LA, USA

Abstract

We define a graph tile to be a unit square (or more generally, a polygon) on which a piece of a graph has been drawn/embedded; in particular, it may have vertices in its interior, edges connecting those vertices, or half-edges that extend to the boundary of the tile. In a graph tiling problem, we are given as input a set of graph tiles, with multiplicities, and the output is an arrangement of those tiles forming a graph of larger area. We focus on a simple tile set: unit square tiles with a central vertex and either a half-edge or no half-edge on each side. Up to symmetry this gives us six different types. We characterize which multiplicities are compatible for sets of at most three different tiles.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases graph tiles

Digital Object Identifier 10.4230/LIPIcs.GD.2025.51

Category Poster Abstract

Acknowledgements This research was initiated at Dagstuhl Seminar 25201, "Computational Geometry". We thank the organizers and all participants for a stimulating atmosphere and valuable discussions. This research was made possible by the Centre for Unusual Collaborations (CUCo) under the project "Going Off the Grid".

1 Introduction

Motivation from material engineering. Irregular architected materials have outstanding mechanical properties, like high stiffness-to-weight ratios and energy absorption, but their rational design is still an open challenge. The recent development of computer-aided virtual growth algorithms (VGAs) has emerged as a new promising venue that allows the generation of architected materials with tailorable mechanical properties [3]. VGAs rely on a specific set of simplified tiles that are assembled on a predefined grid, following a precise set of connectivity rules. Depending on the relative concentration of different tiles, different materials architectures and patterns can be generated, leading in turn to completely different mechanical (and functional) material properties [2, 4].



© Oswin Aichholzer, Robert Ganian, Phillip Keldenich, Maarten Löffler, Gert Meijer, Alexandra Weinberger, and Carola Wenk;
licensed under Creative Commons License CC-BY 4.0



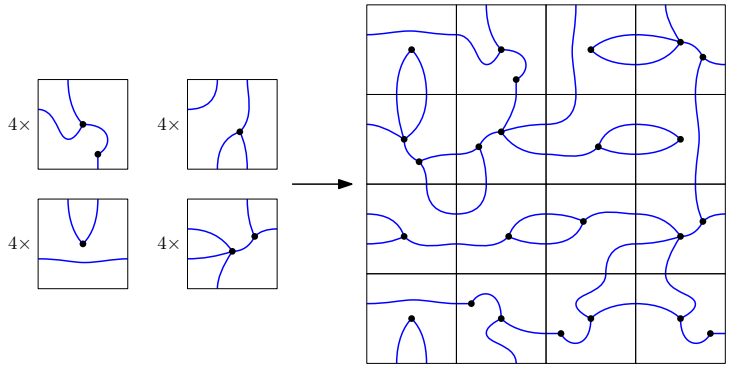


Figure 1 A graph tiling problem and possible solution.

The graph tiling problem. Underlying this engineering challenge lies a mathematical problem. We define a *graph tile* to be a unit square (or more generally, a polygon) on which a piece of a graph has been drawn/embedded; in particular, it may have vertices in its interior, edges connecting those vertices, or half-edges that extend to the boundary of the tile. In a *graph tiling* problem, we are given as input a set of graph tiles, with *multiplicities*, and the output is an arrangement of those tiles into a larger area. Figure 1 shows an example. We can then ask for the output graph to satisfy certain properties, or to optimize over certain criteria, e.g. connectedness, shortest paths, stretch factor, tortuosity, etc. In order to effectively inform VGAs, it is necessary to understand the relation between tile sets and the achievable graph properties in a resulting graph tiling.

Contribution. In this work, we initiate the fundamental study into this area by considering one of the simplest possible sets of tiles: square tiles, with a single vertex in the center, and which do or do not have a half-edge extending to each of the four sides. Taking symmetries into account, this gives us six distinct tiles, see Figure 2. These tiles have been previously identified as the simplest set which is rich enough to encode meaningful material design [4].

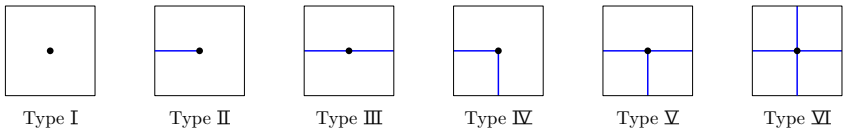


Figure 2 The six different types of tiles considered in this work.

An instance can then be characterized by six integers, the multiplicities of each tile, and a desired output region. Already determining whether the tiles can be arranged with their sides matching is a non-trivial problem, and is related to well-studied topics such as Wang tiles [5], tile self-assembly [1], and more. However, those problems typically do not specify multiplicities of the available tiles, and understanding which multiplicities are compatible is a necessary first step towards solving graph tiling problems. In this work, we characterize which multiplicities are compatible for all sets of at most three different tiles. We restrict our attention to instances in which the total number of tiles is a square number n^2, that need to be assembled into an $n \times n$ grid, and assume input integers are encoded in binary.

Table 1 Results for all sets of at most three distinct tiles. The class with the value "?" contains problems that are compatible and those that are not compatible and we conjecture that there exists a polynomial-time algorithm to distinguish them, but this is still open.

				$\langle \mathrm{I}, \mathrm{II}, \mathrm{III} \rangle \triangleq \langle \mathrm{III}, \mathrm{V}, \mathrm{VI} \rangle$	YES
		$\langle \mathrm{I}, \mathrm{II} \rangle \triangleq \langle \mathrm{V}, \mathrm{VI} \rangle$	YES	$\langle \mathrm{I}, \mathrm{II}, \mathrm{IV} \rangle \triangleq \langle \mathrm{IV}, \mathrm{V}, \mathrm{VI} \rangle$	YES
		$\langle \mathrm{I}, \mathrm{III} \rangle \triangleq \langle \mathrm{III}, \mathrm{VI} \rangle$	P	$\langle \mathrm{I}, \mathrm{II}, \mathrm{V} \rangle \triangleq \langle \mathrm{II}, \mathrm{V}, \mathrm{VI} \rangle$	YES
		$\langle \mathrm{I}, \mathrm{IV} \rangle \triangleq \langle \mathrm{IV}, \mathrm{VI} \rangle$	YES	$\langle \mathrm{I}, \mathrm{II}, \mathrm{VI} \rangle \triangleq \langle \mathrm{I}, \mathrm{V}, \mathrm{VI} \rangle$	?
$\langle \mathrm{I} \rangle \triangleq \langle \mathrm{VI} \rangle$	YES	$\langle \mathrm{I}, \mathrm{V} \rangle \triangleq \langle \mathrm{II}, \mathrm{VI} \rangle$	P	$\langle \mathrm{I}, \mathrm{III}, \mathrm{IV} \rangle \triangleq \langle \mathrm{III}, \mathrm{IV}, \mathrm{VI} \rangle$	YES
$\langle \mathrm{II} \rangle \triangleq \langle \mathrm{V} \rangle$	YES	$\langle \mathrm{I}, \mathrm{VI} \rangle$	NO	$\langle \mathrm{I}, \mathrm{III}, \mathrm{V} \rangle \triangleq \langle \mathrm{II}, \mathrm{III}, \mathrm{VI} \rangle$	P
$\langle \mathrm{III} \rangle$	YES	$\langle \mathrm{II}, \mathrm{III} \rangle \triangleq \langle \mathrm{III}, \mathrm{V} \rangle$	YES	$\langle \mathrm{I}, \mathrm{III}, \mathrm{VI} \rangle$	P
$\langle \mathrm{IV} \rangle$	YES	$\langle \mathrm{II}, \mathrm{IV} \rangle \triangleq \langle \mathrm{IV}, \mathrm{V} \rangle$	YES	$\langle \mathrm{I}, \mathrm{IV}, \mathrm{V} \rangle \triangleq \langle \mathrm{II}, \mathrm{IV}, \mathrm{VI} \rangle$	YES
		$\langle \mathrm{II}, \mathrm{V} \rangle$	YES	$\langle \mathrm{I}, \mathrm{IV}, \mathrm{VI} \rangle$	P
		$\langle \mathrm{III}, \mathrm{IV} \rangle$	YES	$\langle \mathrm{II}, \mathrm{III}, \mathrm{IV} \rangle \triangleq \langle \mathrm{III}, \mathrm{IV}, \mathrm{V} \rangle$	YES
				$\langle \mathrm{II}, \mathrm{III}, \mathrm{V} \rangle$	YES
				$\langle \mathrm{II}, \mathrm{IV}, \mathrm{V} \rangle$	YES

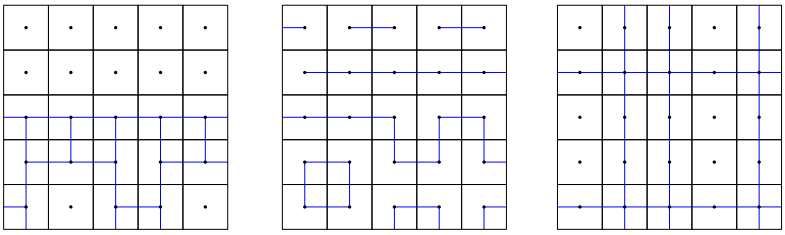


Figure 3 Example tilings for classes $\langle \mathrm{I}, \mathrm{V} \rangle$, $\langle \mathrm{II}, \mathrm{III}, \mathrm{IV} \rangle$, and $\langle \mathrm{I}, \mathrm{III}, \mathrm{VI} \rangle$.

2 Results

We consider the setting where we need to fill an $n \times n$ grid with pre-specified numbers of tiles of each type. We will use the notation $\mathrm{I}, \mathrm{II}, \ldots, \mathrm{VI}$ to refer to the tiles in Figure 2. Note that, apart from rotational symmetry, the tiles also have another type of symmetry: if we replace all sides with half-edges by sides without half-edges and vice versa, we arrive at an equivalent *dual* problem (for the purpose of testing compatibility). We will write $\mathrm{I} \triangleq \mathrm{VI}$, $\mathrm{II} \triangleq \mathrm{V}$, $\mathrm{III} \triangleq \mathrm{III}$, and $\mathrm{IV} \triangleq \mathrm{IV}$ to denote this duality.

We will use the notation $\langle x_1, x_2, \ldots, x_k \rangle$ for the problem class of determining whether a nonzero number of tiles of each type x_1, x_2, to x_k (for $1 \leq k \leq 6$) can be assembled into an $n \times n$ square grid. Each problem class may have value YES (if all problems in the class are compatible), NO (if none of the problems in the class are compatible), P (if compatible and incompatible problems both exist in the class, but it is possible to distinguish them in polynomial time), or NP (if it is NP-hard to distinguish them). Our results are summarized in Table 1. Some interesting cases can be seen in Figure 3.

3 Future Work

We have characterized which multiplicities of at most three tile types are compatible. The clear next step is to extend this classification to classes with up to six different tile types.

Once this is understood, we can begin to investigate graph properties that may be achieved for specific instances, such as obtaining a connected graph, or a tree, or minimizing the area of the largest face / maximizing the area of the smallest face, etc.

References

1 Erik D. Demaine, Martin L. Demaine, Sandor P. Fekete, Mashhood Ishaque, Eynat Rafalin, Robert T. Schweller, and Diane Souvaine. Staged self-assembly: Nanomanufacture of arbitrary shapes with O(1) glues, 2008. arXiv:0803.0316.

2 Chelsea Fox, Kyle Chen, Micaela Antonini, Tommaso Magrini, and Chiara Daraio. Extracting geometry and topology of orange pericarps for the design of bioinspired energy absorbing materials, 2024. (Accepted to Advanced Materials). URL: https://arxiv.org/abs/2404.13351.

3 Ke Liu, Rachel Sun, and Chiara Daraio. Growth rules for irregular architected materials with programmable properties. *Science*, 377:975–981, 2022. doi:10.1126/science.abn1459.

4 Tommaso Magrini, Chelsea Fox, Adeline Wihardja, Athena Kolli, and Chiara Daraio. Control of mechanical and fracture properties in two-phase materials reinforced by continuous, irregular networks. *Advanced Materials*, 36(6):2305198, 2024. doi:10.1002/adma.202305198.

5 Hao Wang. Proving theorems by pattern recognition — II. *The Bell System Technical Journal*, 40(1):1–41, 1961. doi:10.1002/j.1538-7305.1961.tb03975.x.

Graph Tiles

Oswin Aichholzer Robert Ganian Phillip Keldenich Maarten Löffler
Gert Meijer Alexandra Weinberger Carola Wenk

The Problem

Problem statement. Given a set of distinct **graph tiles** and an integer **multiplicity** for each tile, is it possible to arrange all tiles into a square grid such that all tile sides match?

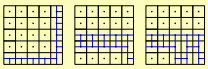
Tile types. In this work, we focus on a simple tile set: unit square tiles with a single vertex in the center, and either a half-edge or no half-edge to each of the four sides. Up to symmetry this gives us six different possibilities.



Type classes. A **class** consists of a subset of the six tile types, e.g. I-II-III. An **instance** of a class is given by a specific multiplicity for each tile type in the class. An instance is **compatible** if it can tile the square.

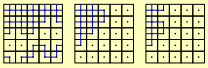
Case highlight: I-V. Perhaps the most interesting class with two tile types is I-V (and its dual II-VI). Given a tiles of type I and b tiles of type V, it is possible to tile an $n \times n$ square if and only if $b = n$ or $b \geq 2n - 1$.

The main idea is that as soon as there is one tile of type V, we must have at least n of them, as we need to fill at least one row or column. A single row is possible when we place it along the edge of the grid. Otherwise, between n and $2n$, the only option is to fill two adjacent edges. As soon as we have two full rows of type V tiles, we can place them in a "ladder" configuration, and then we can selectively add more as needed.



Case highlight: I-IV-VI. One interesting class with three tile types is I-IV-VI (which is dual to itself). Given a tiles of type I , b tiles of type IV, and c tiles of type VI, it is possible to tile an $n \times n$ square if and only if $c \geq n$ or $a \leq \frac{b(b-1)}{2}$ or $c \leq \frac{b(b-1)}{2}$.

The main observation here is that tiles of type I and VI can never be directly adjacent, so they must be separated by a layer of type IV tiles. This separating layer must be at least one tile "thick". If there are enough tiles of type IV, we can separate two regions of any area (if there are too many, we need to take some care how to place the remaining type IV tiles, but this is always possible). If there are not enough, we can only separate a small triangular region from the remainder.



This research was made possible by

CENTRE FOR UNUSUAL COLLABORATIONS

and

SCHLOSS DAGSTUHL Leibniz-Zentrum für Informatik

Motivation. This problem arises in material engineering, where tilings underlie computer-aided algorithms for generating architected materials with tunable properties.

By varying the multiplicities of tile types, structural properties can be changed gradually.

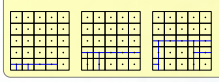
Our set of six tile types has been identified as the simplest set that is suitable for this purpose.

Duality. Note that for any tile set there is also a dual tile set: if we replace all half-edges by non-half-edges and vice versa, we achieve another instance that is compatible if and only if the original one is. Therefore, we only need to consider roughly half of all classes.

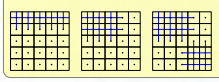
Case highlight: I-III-V. Another interesting class is I-III-V (which is dual to II-III-VI). Given a tiles of type I, $b = kn + r$ (for $r < n$) tiles of type III, and c tiles of type V, it is possible to tile an $n \times n$ square if and only if $X = 1$, where

$$X = \begin{cases} 1, & \text{if } b + c \equiv 0 \mod n, \\ 0, & \text{else if } b + c < n, \\ 1, & \text{else if } c = 1 \text{ and } r < n - k, \\ 0, & \text{else if } c = 1, \\ 0, & \text{else if } b < n \text{ and } c < 2(n - b), \\ 1, & \text{otherwise.} \end{cases}$$

The idea here is that the cases $b + c < n$ and $b + c > 2n$ are not too hard, and for the rest, solutions must follow a very specific structure, with a single full column and limited possibilities in two of the corners.



Case highlight: I-II-VI. The only class with three tile types for which we have not fully classified exactly when it is possible to tile an $n \times n$ square is I-II-VI (and its dual I-V-VI). We conjecture that a polynomial-time algorithm exists to decide, for any given instance, whether a tiling is possible.



Outlook. Next steps are to extend our results to all classes with up to six different tiles. One main question is whether all variants are in P, or if testing compatibility already becomes NP-hard.

Once this is understood, we can investigate properties of the resulting embedded graphs; for instance, testing whether we can make the graph connected, control the sizes of its faces, and more. These properties are crucial in the material engineering application.

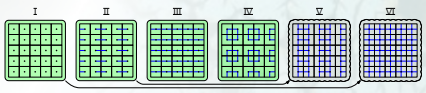
One challenge is that we lose duality when we consider graph properties: the dual of a connected graph is not necessarily connected.

The Results

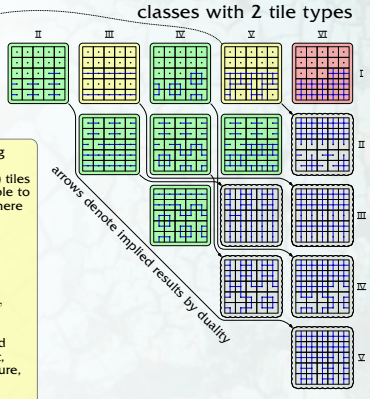
Classification. We classify almost all tile sets with up to 3 different types, giving closed-form formulas to decide the compatibility of any instance, except for one (I-II-VI), which is left for future work.

Note that the classes with 3 types that include neither I nor II are not listed; these are all dual to a type that is listed.

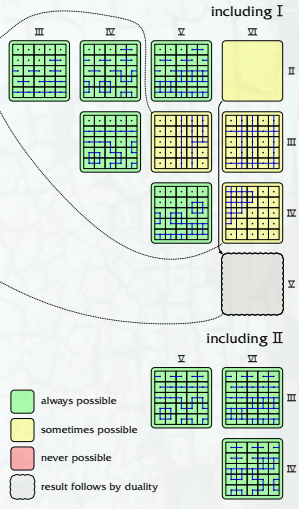
classes with 1 tile type



classes with 2 tile types



classes with 3 tile types



Investigating Crossing Perception in 3D Graph Visualisation

Ying Zhang
University of Konstanz, Germany

Niklas Gröne
University of Konstanz, Germany

Giuseppe Liotta
University of Perugia, Italy

Falk Schreiber
University of Konstanz, Germany
Monash University, Melbourne, Australia

Karsten Klein
University of Konstanz, Germany

Abstract

Human perception and understanding of graph drawings is influenced by a variety of impact factors for which quality measures such as the number of crossings are used as a proxy indicator. For the more and more common stereoscopic 3D (S3D) graph visualisations, evidence is required to better understand graph perception and its relation to quality measures. We investigate the perception of crossing configurations in S3D graph visualisations and present the results of a study.

2012 ACM Subject Classification Human-centered computing → Empirical studies in visualization; Human-centered computing → Virtual reality; Human-centered computing → Graph drawings; Human-centered computing → Visual analytics

Keywords and phrases Graph Perception, Stereoscopic 3D Graph Visualisation, Crossing Configurations

Digital Object Identifier 10.4230/LIPIcs.GD.2025.52

Category Poster Abstract

Related Version *Preprint*: https://doi.org/10.48550/arXiv.2508.00950

Funding We acknowledge funding by DFG under Germany's Excellence Strategy – EXC 2117 – 422037984, and DFG project ID 251654672 – TRR 161; we also acknowledge funding by MUR PRIN Project no. 2022TS4Y3N –1405 EXPAND and by MUR PON Project ARS01 00540 RASTA.

1 Motivation

The investigation of impact factors for graph perception is important to evaluate and improve quality measures and drawing algorithms as well as to better understand human graph perception. The number of edge crossings in a 2D graph drawing has long been a main quality measure for drawing evaluation and is still regularly investigated and reported. Their impact is however dependent on further factors such as graph size or crossing angle [6, 3, 9]. The use of stereoscopic 3D graph visualisations has gained attraction over the last years, and results from several studies indicate that they can improve analysis efficiency for a range of analysis scenarios [4, 1, 2, 7, 10]. The importance and effect of impact factors for perception might differ from the classical 2D setting, e.g. due to the influence of depth cues and eye depth focus. While edge crossings can also occur in 3D, there are edge configurations in space that are not crossings but might be perceived as such from a specific viewpoint.


© Ying Zhang, Niklas Gröne, Giuseppe Liotta, Falk Schreiber, and Karsten Klein;
licensed under Creative Commons License CC-BY 4.0



Such configurations create crossings when projected on the corresponding 2D image plane and could impact readability similar to 2D crossings. To indicate that most of such edge configurations are not edge crossings in 3D per se, but could be perceived as such, we call them '*projective edge crossing configurations*', or 'PECC' in short. The additional depth aspect and the subsequent impact factor of edge distance in space might further influence the importance of those configurations for readability, and also change the influence of aspects such as the perceived crossing angle. We investigate the impact of such factors in an empirical study and report on findings of difference between major factor categories.

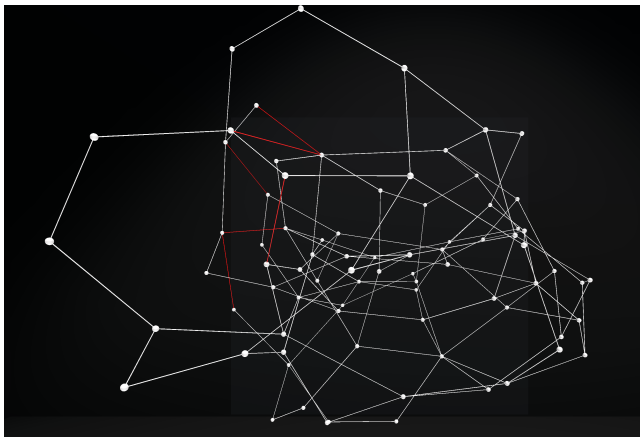


Figure 1 Graph representation for our study in VR, edges highlighted for memorisation task.

2 Study Design and Setup

As an initial step to investigate the effect of PECCs on graph perception in S3D, we investigate their influence on task performance. As controlled impact factors that serve as independent variables for our study, we add the depth distance of the edges in 3D space at the projected crossing points (IV3) to the established criteria of number of crossings (IV1) and crossing angle (IV2) (which we measure in the projection as an indicator of the perceived number and angles). The study participants were asked to perform three established tasks on 3D graph visualisations in VR: shortest path (SP), common neighbours (CN), and memorisation (ME). We recorded the completion time and accuracy, and the participants were told to perform tasks as accurately and as quickly as they could, while the pace was controlled by the participant. We chose a VR environment for the study, as we could fully control the appearance of the environment and graphs to minimise distraction. The participants remained seated during the study, wearing a head-movement tracking VR HMD (Meta Quest 2), with graph visualisations shown centred in front of their viewpoint, see Figure 1. Except for small head movements, there is no interaction that allows the viewer to change the visual representation of the graphs, in order to avoid changes in the impact factor values. We implemented a Unity application based on the VR graph analysis framework GAV-VR [5].

For the stimuli, we restricted our study to a single graph generator model, the Newman-Watts-Strogatz random graph model, which allows to create graphs with a good combination of local clustering and shortest path lengths as often found in practice [8]. For the layout we decided to use a standard spring-model layout, which provides a reasonable readability for the tasks, as tested by us before the study. In order to test clearly separable conditions for our independent variables, the stimuli were categorised into factor levels as having either low

or high number of (projection) crossings - up to 2 (level `L`) and at least one per task-related edge on average (level `H`), crossing angles - below 30° (level `L`) and above 60° (level `H`), and crossing point distances - as fraction of full graph representation depth (up to 0.1 level `L` and above 0.3 level `H`). We calculate these variables task-related: Instead of measuring them across the whole graph, we investigate PECCs that are involving task-relevant edges, i.e. (shortest) path edges for SP and ME and edges adjacent to the selected nodes for CN. This results in eight conditions, encoded as `XYZ` in the following, where `X` indicates the level for IV1 (`H` for high, `L` for low), `Y` the level of the average IV2, and `Z` the level of the average IV3. We recruited 28 participants (12 male, 16 female) at our university, aged 19-36 years (avg. 24.46). Thirteen of them reported prior VR experience, and seven reported previous graph analysis experience. All participants reported normal or corrected-to-normal vision. We performed a within-subjects experiment and counter-balanced task and condition order.

3 Results

We assumed that, in line with previous studies in 2D setups, it will be easier to perform the tasks with less crossings and larger angles, and in addition that larger edge distance would be the easier factor level, too. In summary, we have for accuracy significant differences between high and low PECC numbers (all tasks), but only for task SP between high and low angles and between high and low distances. For completion time (see Table 1), we recorded significantly shorter times for low PECC number combinations (`LXX`) across all tasks, shorter times for low angle combinations (`XLX`) for tasks SP and CN, and no significant differences for distance levels. We could see significant impact on both accuracy and completion time based on differences between pairs of conditions, but also between low and high levels of the number of PECC. Compared to known 2D results, we could observe similar effects for the number, but in contrast to 2D results no similar result for angles, which we deem an important find to be further investigated in future research. For depth impact, we established significant differences between high and low levels of PECC distance. Thus, we think that a deeper investigation of PECC impact factors should be a topic of future research to improve layout quality measures and algorithms for S3D.

Table 1 Accuracy and mean RT across all tasks per condition, sorted by RT. Expected impact of count, angle, distance indicated based on our hypothesis. Highest and lowest values in bold.

CC	Count	Angle	Distance	Accuracy (%)	Mean RT (s)
HHH	Hard	Easy	Easy	50.0 %	**21.3**
HHL	Hard	Easy	Hard	**42.9** %	20.7
HLL	Hard	Hard	Hard	53.6 %	19.0
HLH	Hard	Hard	Easy	65.5 %	17.1
LHL	Easy	Easy	Hard	73.8 %	15.4
LHH	Easy	Easy	Easy	79.8 %	15.1
LLL	Easy	Hard	Hard	76.2 %	12.7
LLH	Easy	Hard	Easy	**82.1 %**	**12.5**

References

1 Stefan P Feyer, Bruno Pinaud, Stephen Kobourov, Nicolas Brich, Michael Krone, Andreas Kerren, Michael Behrisch, Falk Schreiber, and Karsten Klein. 2D, 2.5 D, or 3D? an exploratory study on multilayer network visualisations in virtual reality. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):469–479, 2023. doi:10.1109/TVCG.2023.3327402.

2 Nicolas Greffard, Fabien Picarougne, and Pascale Kuntz. Beyond the classical monoscopic 3D in graph analytics: An experimental study of the impact of stereoscopy. In *2014 IEEE VIS International Workshop on 3DVis (3DVis)*, pages 19–24, 2014. doi:10.1109/3DVis.2014.7160095.

3 Weidong Huang, Seok-Hee Hong, and Peter Eades. Effects of Crossing Angles. In *2008 IEEE Pacific Visualization Symposium*, pages 41–46, 2008. doi:10.1109/PACIFICVIS.2008.4475457.

4 Lucas Joos, Maximilian T Fischer, Julius Rauscher, Daniel A Keim, Tim Dwyer, Falk Schreiber, and Karsten Klein. Visual network analysis in immersive environments: A survey. *arXiv preprint arXiv:2501.08500*, 2025. doi:10.48550/arXiv.2501.08500.

5 Wilhelm Kerle-Malcharek, Stefan P. Feyer, Falk Schreiber, and Karsten Klein. Gav-vr : an extensible framework for graph analysis and visualisation in virtual reality. In Jean-Marie Normand, editor, *ICAT-EGVE 2023*, pages 131–139. The Eurographics Association, 2023. doi:10.2312/egve.20231321.

6 Stephen G. Kobourov, Sergey Pupyrev, and Bahador Saket. Are crossings important for drawing large graphs? In Christian Duncan and Antonios Symvonis, editors, *Graph Drawing*, pages 234–245. Springer Berlin Heidelberg, 2014. doi:10.1007/978-3-662-45803-7_20.

7 Oh-Hyun Kwon, Chris Muelder, Kyungwon Lee, and Kwan-Liu Ma. A study of layout, rendering, and interaction methods for immersive graph visualization. *IEEE Trans. Vis. Comput. Graph.*, 22(7):1802–1815, 2016. doi:10.1109/TVCG.2016.2520921.

8 M.E.J. Newman and D.J. Watts. Renormalization group analysis of the small-world network model. *Physics Letters A*, 263(4):341–346, 1999. doi:10.1016/S0375-9601(99)00757-4.

9 Helen Purchase. Which aesthetic has the greatest effect on human understanding? In Giuseppe Di Battista, editor, *Graph Drawing*, pages 248–261. Springer Berlin Heidelberg, 1997. doi:10.1007/3-540-63938-1_67.

10 J. Sorger, A. Arleo, P. Kán, W. Knecht, and M. Waldner. Egocentric network exploration for immersive analytics. *Comput. Graph. Forum*, 40(7):241–252, 2021. doi:10.1111/cgf.14417.

Perception of Edge Crossings in 3D

An Investigation of Projective Edge Crossing Configurations in Stereoscopic 3D

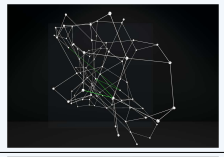
Universität Konstanz
1. University of Konstanz
2. University of Perugia
Ying Zhang[1] ● Niklas Gröne[1] ● Giuseppe Liotta[2] ● Falk Schreiber[1] ● Karsten Klein[1]
Contact: ying.zhang@uni-konstanz.de

The use of stereoscopic 3D (S3D) graph visualisations has gained attraction over the last years, and results from several studies indicate that they can improve analysis efficiency for a range of analysis scenarios [1,2,3,4]. The importance and effect of impact factors for graph perception, as covered by many established quality measures, might differ from the classical 2D setting. The impact of edge crossings has been considerably studied in 2D, but there are no similar investigations for S3D. We conducted a first study and report on our findings.

Perceived Edge Crossing Configurations: There are edge configurations in space that don't need to be crossings but might be perceived as such from a specific viewpoint. Such configurations create crossings when projected on the corresponding 2D image plane and could impact readability similar to (2D) crossings. We call them '**projective edge crossing configurations**', or '**PECC**' in short. Several factors might determine the impact of PECCs, in particular similar to the 2D case their number as well as the crossing angle (in the projection). Further factors are specific to the S3D scenario, such as the distance of the edges in space at the perceived crossing point.

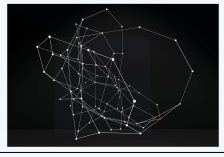
Aim: Investigate the impact of PECCs on task performance, using as independent variables PECC number, PECC crossing angle, and PECC crossing edge distance. For each independent variable we distinguish two levels, low (L) and high (H). This results in eight conditions, represented as XYZ in the following, where X indicates the level the number, Y the level of the average angle, and Z the level of the average distance of PECCs.

Tasks and Procedure: The study participants were asked to perform three established tasks on 3D graph visualisations in VR: shortest path (SP), common neighbours (CN), and memorisation (ME). Only PECCs that are directly related to task-relevant structures are taken into account, e.g. PECCs involving edges of the shortest path. The participants remained seated during the study, wearing a head-movement tracking VR HMD (Meta Quest 2), with graph visualisations shown centred in front of their viewpoint. Except for small head movements, there is no interaction that allows the viewer to change the visual representation of the graphs, in order to avoid changes in the conditions. PECC distance is measured as the ratio of the distance to the overall drawing depth.

Task 1: Shortest path - given two highlighted nodes, find a shortest path between them and report its length (in number of edges).

Left: Challenge stimulus with highlighted nodes
Right: Solution path, with a high number of PECCs (on average at least one per edge), high average PECC angles (greater than 60°) and high average PECC depth distance (condition HHH)

Task 2: Common neighbours - given two highlighted nodes, count their common neighbours and report their number.

Left: Challenge stimulus with highlighted nodes
Right: Common neighbours highlighted in green, with a high number of PECCs on edges adjacent to the red nodes, low average PECC angles (less than 30°) and low average PECC depth distance (condition HLL)

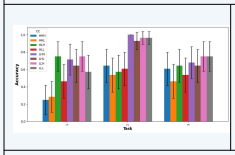



Task 3: Memorisation - given a highlighted path, and a highlighted set of edges shown after a pause, decide the overlap and report the number of edges highlighted twice.

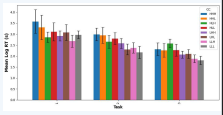
Left: Memorisation stimulus with path highlighted, with low number of PECCs, high average PECC angles, and high average PECC depth distance (difficult to perceive in 2D printing) (condition LHH)
Right: Challenge stimulus with single edges highlighted

(Note: The representation was slightly adjusted for better readability in a 2D print by changing the lighting.)

Results



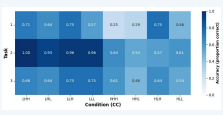
Accuracy and mean log completion time by condition and task.



CC	Count	Angle	Distance	Accuracy (%)	Mean RT (s)
HHH	Hard	Easy	Easy	50.0 %	**21.3**
HHL	Hard	Easy	Hard	**42.9** %	20.7
HLL	Hard	Hard	Hard	53.6 %	19.0
HLH	Hard	Hard	Easy	65.5 %	17.1
LHL	Easy	Easy	Hard	73.8 %	15.4
LHH	Easy	Easy	Easy	79.8 %	15.1
LLL	Easy	Hard	Hard	76.2 %	12.7
LLH	Easy	Hard	Easy	**82.1** %	**12.5**

Expected impact of PECC count, angle, distance indicated. Highest and lowest values in bold.

Accuracy and mean RT across all tasks per condition, sorted by RT.



Heat map for accuracy by condition and task.

Summary: Significant differences in accuracy between high and low PECC numbers (all tasks), but only for task SP between high and low angles and between high and low distances. For completion time, significantly shorter times for low PECC number combinations (LXX) across all tasks, significantly shorter times for low angle combinations (XLX) for tasks SP and CN, and no significant differences for distance levels. Significant impact on both accuracy and completion time based on differences between pairs of conditions, but also between low and high levels of the number of PECC. Compared to known 2D results, we could observe similar effects for the number, but in contrast to 2D results no similar result for angles, which we deem an important find to be further investigated in future research. For depth impact, we established significant differences between high and low levels of PECC depth.

References: [1] Weidong Huang, Seok-Hee Hong, and Peter Eades. Effects of Crossing Angles. In 2008 IEEE Pacific Visualization Symposium, pages 41–46, March 2008. doi:10.1109/PACIFICVIS.2008.4475457.
[2] Wilhelm Kerle-Malcharek, Stefan P. Feyer, Falk Schreiber, and Karsten Klein. GAV-VR: An Extensible Framework for Graph Analysis and Visualisation in Virtual Reality. In Jean-Marie Normand, editor, ICAT-EGVE 2023, pages 131–139. The Eurographics Association, 2023. doi:10.2312/egve.20231321.[3] Stephen G Kobourov, Sergey Pupyrev, and Bahador Saket. Are crossings important for drawing large graphs? In Graph Drawing: 22nd International Symposium, GD 2014, pages 234–245. Springer, 2014. [4] Stefan P Feyer, Bruno Pinaud, Stephen Kobourov, Nicolas Brich, Michael Krone, Andreas Kerren, Michael Behrisch, Falk Schreiber, and Karsten Klein. 2D, 2.5 D, or 3D? an exploratory study on multilayer network visualisations in virtual reality. IEEE Transactions on Visualization and Computer Graphics, 30(1):469–479, 2023.

Edge Bundling as a Multi-Objective Optimization Problem

Raissa dos Santos Vieira
Universidade Federal de Goiás, Goiânia, Brazil

Hugo A. D. do Nascimento
Universidade Federal de Goiás, Goiânia, Brazil

Joelma de Moura Ferreira
IPOG Instituto de Pós-Graduação e Graduação, Goiânia, Brazil

Les Foulds
Universidade Federal de Goiás, Goiânia, Brazil

Karsten Klein
University of Konstanz, Germany

Falk Schreiber
University of Konstanz, Germany
Monash University, Melbourne, Australia

Abstract

Edge bundling is a technique commonly used to reduce visual clutter and improve the comprehension of the drawings of large graphs. Here, we model edge bundling as a multi-objective optimization problem and employ clustering strategies, metaheuristic and Pareto analysis to identify non-dominated solutions for some classical graphs from the literature.

2012 ACM Subject Classification Computing methodologies

Keywords and phrases Graph Drawing, Edge Bundling, Visual Clutter, Multi-objective Optimization

Digital Object Identifier 10.4230/LIPIcs.GD.2025.53

Category Poster Abstract

Funding H. A. D. do Nascimento, K. Klein, and F. Schreiber acknowledge funding by DFG under Germany's Excellence Strategy – EXC 2117 – 422037984, and DFG project ID 251654672 – TRR 161.

Acknowledgements We thank the computer support from LaMCAD/UFG, Brazil.

1 Introduction

A large variety of methods exist that realize edge bundling, see e. g. [3, 7, 4]. A formalization of some explicit edge bundling optimization problems has been proposed in [1]. Among those problems, the General-Based Edge Bundling (GBEB) Problem was introduced that aims at minimizing the number of bundles and at maximizing a given set of compatibility measures. The authors investigated in [1] a specific version of the GBEB, which consists of maximizing a weighted sum of two objective components: 1) the reciprocal of the number of bundles; and 2) the sum of the compatibility value for every bundle, calculated using the product of distinct compatibility measures for each pair of edges grouped together and a threshold to regulated the signal of the component. An evolutionary algorithm for the problem was designed and achieved good results. In a further work presented in [6], the authors treated the GBEB as a clustering problem and employed a combination of clustering and clustering-ensemble methods to solve it. The new approach led to better solutions than



© Raissa dos Santos Vieira, Hugo A. D. do Nascimento, Joelma de Moura Ferreira, Les Foulds, Karsten Klein, and Falk Schreiber;
licensed under Creative Commons License CC-BY 4.0



when using the evolutionary algorithm. However, both studies adopted an objective function that combines distinct objectives in a fixed way, thus limiting the space of possibly desirable solutions.

This paper investigates the GBEB problem as a multi-objective optimization problem, by searching for the Pareto frontier considering all objectives independently, instead of combining them in a single objective function. The multi-objective version of GBEB (abbreviated here as MEB) is still treated as a clustering problem, as done in [6], but we go beyond the previous methodological approaches: a step-by-step method is adopted that combines bundling strategies with a meta-heuristic to extensively explore the solution space and find good compromise solutions.

2 The Multi-objective Approach

We aim at finding clustering solutions that maximize seven distinct aspects: 1) the reciprocal of the number of edge bundles (number of edge clusters), 2)-5) four compatibility measures proposed by Holten in [3] (angle, scale, position and visibility compatibilities), 6) the distance compatibility presented in [1], and 7) a new topological compatibility measure, which is defined as the distance between edge embeddings obtained with the *node2vec* method [2]. Note that only aspects 2) to 7) are compatibility measures. Here, the measurements in every quality aspect are normalized with maximum value =1.0.

The well known concept of *dominance* is used for comparing clustering solutions. A set of solutions can be divided into a hierarchical sequence of *dominance classes* by the following recursive procedure: solutions not dominated by any other solution in the set comprises the first class (called the *non-dominated class*); then, these solutions are removed from the set and the process repeats for the remaining set. While a class contains only equivalent solutions, any class is considered better than the subsequent classes. The first class constitutes the *Pareto frontier*. Our optimization method inputs a given graph $G = (V, E)$ and a straight-line drawing D of G, and performs the following three steps, where the first two are similar to what was done in [6]:

- **Step 1 – Creating edge representations**: Given G and D, an edge-to-edge compatibility matrix (dimension $|E| \times |E|$) is created for every compatibility measure. The i-th line of the j-th matrix, with $1 \leq i \leq |E|$ and $1 \leq j \leq 6$, is the *feature representation* of edge i according to the compatibility measure j.
- **Step 2 – Producing first solutions via clustering algorithms**: Four standard clustering algorithms (Agglomerative Hierarchical, K-Means, K-Means with Mini Batch, and Spectral Clustering) with various parameter configurations are employed to cluster the graph edges for each compatibility measure individually, using their related edge feature representation. A normalized quality vector (with seven values, representing the measurement for seven previously mentioned quality aspects) is computed for every solution. Finally, the 1000 solutions with the quality vector closest to $[1, 1, 1, 1, 1, 1, 1]$ are taken to the next step.
- **Step 3 – Improving solutions via multi-objective consensus**: This step runs an Asynchronous Team (A-Team) of autonomous agents [5] to improve the clustering solutions generated in Step 2, as an evolving group of dominance classes. The A-Team architecture consists of a shared memory with a predefined number N, of clustering solutions, and eighteen agents, which were formed by three clustering consensus algorithms with different setup parameters. Each agent reads up to k solutions randomly chosen from the memory (k is also predefined), runs a clustering consensus strategy, and writes

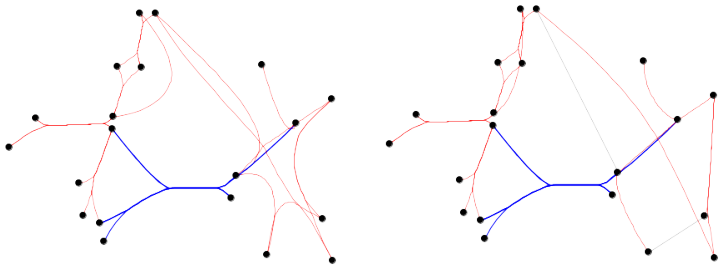
its resultant solution to the shared memory. Every new solution produced may cause the dominance classes to be recalculated and an earlier solution (preferably from the latest dominance classes) to be removed. The A-Team runs until a pre-specified stop condition is reached, when the non-dominated class of solutions (the final Pareto frontier) is outputted.

The solutions in the Pareto frontier can be graphically rendered by using the adapted force-directed edge bundling method described in [1]. It takes the graph, the position of its vertices and a list of sets of edges (clusters) and produces an edge-bundling graph drawing.

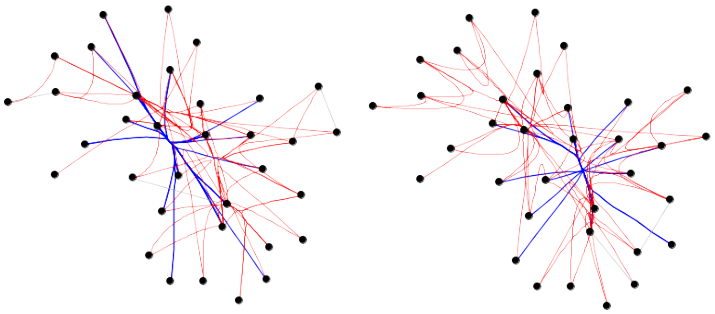
3 Experiments and Results

In order to validate the new multi-objective optimization approach, experiments were performed with five sparse graphs from the literature, with sizes ranging from 20 to 160 vertices, and from 28 to 161 edges. We also set $N = 1000$ and $k = 10$ in Step 3, and ran the A-Team algorithm for one hour (this was sufficient for producing good quality solutions).

As the main result, the A-Team algorithm was capable of improving the set of solutions by finding new non-dominated ones. It also reduced the size of the non-dominated class for all graphs, demonstrating that many solutions generated in Step 2 could be replaced by a few better ones. Another important contribution of the general approach is the diversification of compromise solutions, with regard to the distinct quality aspects, which becomes ready for human evaluation, as illustrated in Figure 1 (blue lines indicate the largest bundle).



(a) Synthetic graph



(b) Zachary Karate Club graph

Figure 1 Examples of compromise solutions in the Pareto frontier for two of the five graphs.

References

1 Joelma De M. Ferreira, Hugo A. D. Do Nascimento, and Les R. Foulds. An evolutionary algorithm for an optimization model of edge bundling. *Information*, 9(7), 2018. doi:10.3390/info9070154.

2 Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653, 2016. doi:10.48550/arXiv.1607.00653.

3 Danny Holten and Jarke J. van Wijk. Force-directed edge bundling for graph visualization. *Comput. Graph. Forum*, 28(3):983–990, 2009. doi:10.1111/j.1467-8659.2009.01450.x.

4 Quan Hoang Nguyen, Seok-Hee Hong, and Peter Eades. TGI-EB: A new framework for edge bundling integrating topology, geometry and importance. In Marc J. van Kreveld and Bettina Speckmann, editors, *Graph Drawing - 19th International Symposium, GD 2011, Eindhoven, The Netherlands, September 21-23, 2011, Revised Selected Papers*, volume 7034 of *Lecture Notes in Computer Science*, pages 123–135. Springer, Springer, 2011. doi:10.1007/978-3-642-25878-7_13.

5 S Talukdar, L Baerentzen, A Gove, and P De Souza. Asynchronous teams: Cooperation schemes for autonomous agents. *Journal of heuristics*, 4(4):295–321, 1998. doi:10.1023/A:1009669824615.

6 Raissa S. Vieira, Hugo A. D. do Nascimento, Joelma de Moura Ferreira, and Les R. Foulds. Clustering ensemble-based edge bundling to improve the readability of graph drawings. In *26th International Conference Information Visualisation, IV 2022, Vienna, Austria, July 19-22, 2022*, pages 21–26. IEEE, 2022. doi:10.1109/IV56949.2022.00013.

7 Markus Wallinger, Daniel Archambault, David Auber, Martin Nöllenburg, and Jaakko Peltonen. Faster edge-path bundling through graph spanners. *Comput. Graph. Forum*, 42(6):e14789, 2023. doi:10.1111/cgf.14789.

Edge Bundling as a Multi-objective Optimization Problem

Raissa dos Santos Vieira [1] Hugo A. D. do Nascimento [1] Joelma de Moura Ferreira [2]
Les Foulds [1] Karsten Klein [3] Falk Schreiber [3]

[1]Universidade Federal de Goiás, Brazil [2]IPOG Instituto de Pós-Graduação e Graduação, Brazil [3]Konstanz University, Germany

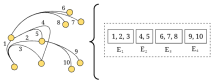
Introduction

Edge bundling is a technique commonly used to reduce visual clutter by grouping edges in bundles.



General-Based Edge-Bundling (GBEB)

Consists of an explicit edge bundling optimization problem that aims to minimize the number of bundles, and to maximize a set of compatibility measures. Solved using an evolutionary algorithm [1].



Clustering Ensemble-Based Edge Bundling Approach

Treats GBEB as a clustering problem, performing edge bundling through edge-clustering strategies followed by a clustering ensemble [2].

Max. $f = w_1 \times \frac{1}{n} + w_2 \times C_G$
with n = number of bundles, T a threshold, and $C_G = \sum_{i=1}^{n} C_{E_i}$

$$C_{E_i} = \begin{cases} \sum C(p,q), \forall p,q \in E_i & \text{if } \min(C(p,q)) \geq T; \\ p_n & \text{if } \min(C(p,q)) < T; \\ 0, & \text{if } |E_i| = 1. \end{cases}$$

$$P_i = \begin{cases} \sum C(p,q) \cdot (-3), & \text{if } \sum C(p,q) > 0; \\ -3, & \text{if } \sum C(p,q) = 0. \end{cases}$$

$$C(p,q) = C_a(p,q) \times C_s(p,q) \times C_p(p,q) \times C_v(p,q) \times C_d(p,q)$$

Multi-objective Edge Bundling (MEB)

MEB is a variant of GBEB defined as:

Let $G = (V, E)$ be a simple graph and D a straight-line drawing of G. Let $C_1(p,q), C_2(p,q), ..., C_m(p,q)$ be m compatibility measures defined for every pair of edges $p, q \in E$ based on D. The multi-objective explicit edge bundling problem in decentralized bundles (MEB) consists of determining a decomposition of E into disjoint subsets $E_1, E_2, ..., E_n$, with $E = \cup_{i=1}^{n} E_i$ and $E_i \cap E_j \neq \emptyset$ for $1 \leq i, j \leq |E|$ and $i \neq j$, that maximizes the objective vector $\overline{F} = [F_B, C_1, C_2, ..., C_m]$, where $F_B = \frac{1}{n}$ and $C_k = \sum_{i=1}^{n} C_{k,E_i}$ for $k = 1, 2, \ldots, m$, with C_{k,E_i} a version of C_{E_i} that uses only the k-th compatibility measure.

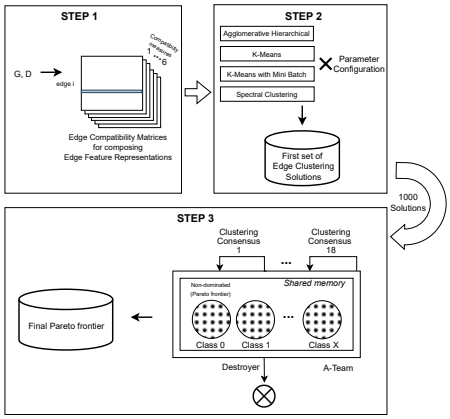
Additional explanation:

- The goal is to find a decomposition of E into disjoint subsets (clusters or bundles) that maximizes the values of $\overline{F}$.
- The number n of subsets of E is variable, with $1 \leq n \leq |E|$.
- Maximizing the vector $\overline{F}$ involves:
  1. Minimizing the number of bundles (n); and
  2. Maximizing m predefined compatibility measures.
- For the current study, $m = 6$ and the compatibility measures are *angle*, *scale*, *position*, *visibility*, *distance*, and *topological*.

Clustering Ensemble for Multi-objective Edge Bundling

To solve the multi-objective edge bundling problem, we present an approach that uses the edge-clustering strategies adopted in [2], but extend it to support Pareto Analysis and to include an Asynchronous Team (A-Team) algorithm.

The approach consists of the following steps:



Experiments and Results

Experiments were performed with five graphs, named G1 to G5. For the A-Team, we set $N = 1000$ as its memory size, and $k = 10$ as the maximum number of solutions for selection by any consensus agent. The A-Team ran for 3600 seconds for each graph.

Table 1 presents information about the graphs and the number of solutions in the Pareto frontier at Step 3. The evolution of the number of solutions on the Pareto frontier during the experiments can be observed in Figure 1. Figure 2 shows bundling solutions from the final Pareto frontier.

Graph	|V|	|E|	Pareto Frontier *Initial*	Pareto Frontier *Final*	No. of Iterations	New Solutions in Pareto Frontier
G1 – Synthetic	20	28	325	35	22698	28
G2 – Zachary Karate Club	34	78	828	262	1908	16
G3 – Planar Graph GD2015	66	101	882	451	1260	20
G4 – Dolphin Social Network	62	159	958	903	198	65
G5 – MovieLens	160	161	758	688	198	16

Table 1. Graph attributes and the number of solutions in the Pareto frontier in Step 3 of the approach.

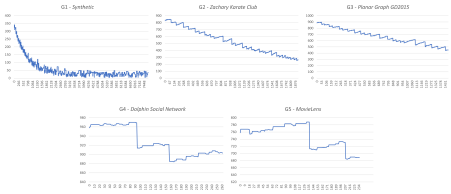
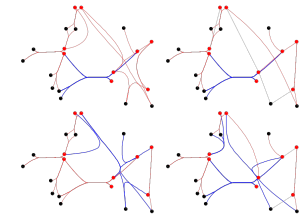
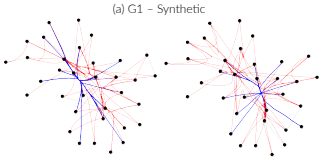


Figure 1. The change in the number of solutions in the Pareto frontier.



(a) G1 – Synthetic



(b) G2 – Zachary Karate Club

Figure 2. Examples of compromise solutions in the Pareto frontier for G1 and G2.

Main Results:

- A-Team was capable of improving the set of solutions by finding new non-dominated ones;
- The size of the non-dominated class for all graphs largely reduced over time, showing that many solutions generated in Step 2 could be replaced by better ones;
- The overall approach produces a set of compromise edge bundling solutions, with regard to the distinct quality aspects, thus supporting further human evaluation.

References

[1] Joelma De M. Ferreira, Hugo A. D. Do Nascimento, and Les R. Foulds. An evolutionary algorithm for an optimization model of edge bundling. *Information*, 9(7), 2018.

[2] Raissa dos Santos Vieira, Hugo A. D. do Nascimento, Joelma Moura Ferreira, and Les R. Foulds. Clustering ensemble-based edge bundling to improve the readability of graph drawings. In *Proceedings of 26th International Conference on Information Visualisation*, pages 21–26, 2022.

BH-tsNET, FIt-tsNET, L-tsNET: Fast tsNET Algorithms for Large Graph Drawing

Amyra Meidiana
The University of Sydney, Australia

Seok-Hee Hong
The University of Sydney, Australia

Kwan-Liu Ma
University of California at Davis, CA, USA

Abstract

The `tsNET` algorithm adapts the popular dimensional reduction method `t-SNE` for graph drawing to compute high-quality drawings, preserving the neighborhood and clustering structure. However, its $O(nm)$ runtime results in poor scalability for large graphs. In this poster, we present three fast algorithms for reducing the time complexity of `tsNET` to $O(n \log n)$ time and $O(n)$ time, by integrating new fast methods for computation of high-dimensional probabilities and entropy computation with fast `t-SNE` algorithms for computation of KL divergence gradient. Specifically, we present two $O(n \log n)$-time algorithms `BH-tsNET` and `FIt-tsNET`, incorporating partial BFS-based high-dimensional probability computation and a new quadtree-based entropy computation with fast `t-SNE` algorithms, and $O(n)$-time algorithm `L-tsNET`, introducing a new fast interpolation-based entropy computation. Extensive experiments using benchmark data sets confirm that `BH-tsNET`, `FIt-tsNET`, and `L-tsNET` outperform `tsNET`, achieving 93.5%, 96%, and 98.6% faster runtime, respectively, while computing similar quality drawings in terms of quality metrics (neighborhood preservation, stress, shape-based metrics, and edge crossing) and visual comparison.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases tsNET, t-SNE, Large Graph Drawing

Digital Object Identifier 10.4230/LIPIcs.GD.2025.54

Category Poster Abstract

1 Introduction

The `tsNET` [3] algorithm for graph drawing utilizes the popular dimension reduction method `t-SNE` *(t-Stochastic Neighborhood Embedding)* [11], which aims to preserve the *neighborhood* of data points in a low-dimensional projection.

`t-SNE` models the distances of data points in the high- and low-dimensional spaces as probability distributions. A low-dimensional projection is then computed by minimizing the *KL (Kullback-Leibler) divergence* [4] between the two probability distributions, typically using gradient descent, by moving the points in the projection according to the gradient of the KL divergence. The runtime of `t-SNE` is $O(n^2)$, due to the computation of pairwise distances between all pairs of data points.

`tsNET` uses the *graph theoretic distance* (i.e., shortest path) between vertices in a graph $G = (V, E)$ as the high-dimensional distance, which takes $O(nm)$ time, where $n = |V|$ and $m = |E|$. Specifically, `tsNET` adds the following two more terms to the cost function, on top of the KL divergence term used by `t-SNE`: an $O(n)$-time *compression* term to accelerate untangling the drawing, and an $O(n^2)$-time *entropy* term for reducing clutter.

`tsNET` computes high-quality graph drawings which preserve the clustering and neighborhood structure of vertices. However, it is not scalable to large graphs due to the $O(nm)$ runtime for all-pair shortest path computation and $O(n^2)$ runtime for the cost function.


© Amyra Meidiana, Seok-Hee Hong, and Kwan-Liu Ma;
licensed under Creative Commons License CC-BY 4.0



Recently, fast `t-SNE` algorithms have been introduced. For example, `BH-SNE` *(Barnes-Hut SNE)* [10] utilizes kNN (k-Nearest Neighbors) and quadtree to reduce the computation of high-dimensional probabilities and KL divergence gradient to $O(n \log n)$ time. More recently, `FIt-SNE` *(FFT-accelerated Interpolation-based t-SNE)* [5] utilizes $O(n \log n)$-time approximate kNN to compute the high-dimensional probabilities, and a FFT (Fast Fourier Transform) [8]-accelerated interpolation to further improve the runtime of the KL divergence gradient computation to $O(n)$ time.

However, algorithms reducing the overall runtime of `tsNET` have not been investigated yet. To reduce the runtime of `tsNET`, there are three $O(n^2)$-time components that need to be reduced: computation of all pair shortest paths, computation of KL divergence gradient, and entropy computation. While it was mentioned as a possible future work to use fast `t-SNE` algorithms to improve the runtime of the KL divergence term [3], methods to reduce the runtime of the *entropy* term were not considered yet.

In this poster, we present three new fast `tsNET` algorithms for drawing large graphs, integrating new fast methods for all-pair shortest path computation and entropy computation, with fast `t-SNE` algorithms. Specifically, our fast algorithms use a new $O(n)$-time partial BFS method for all-pair shortest path computation.

1. The first $O(n \log n)$-time `BH-tsNET` algorithm uses a new $O(n \log n)$-time quadtree-based entropy computation, integrated with $O(n \log n)$-time quadtree-based KL divergence computation of `BH-SNE`.
2. Faster $O(n \log n)$-time `FIt-tsNET` algorithm integrates the $O(n \log n)$-time quadtree-based entropy computation with $O(n)$-time interpolation-based KL divergence computation of `FIt-SNE`.
3. The fastest $O(n)$-time `L-tsNET` algorithm uses a new $O(n)$-time FFT-accelerated interpolation-based entropy computation, integrating $O(n)$-time interpolation-based KL divergence computation of `FIt-SNE`.

Table 1 summarizes the time complexity analysis of our fast algorithms. For the technical details, see the full version of this poster [7].

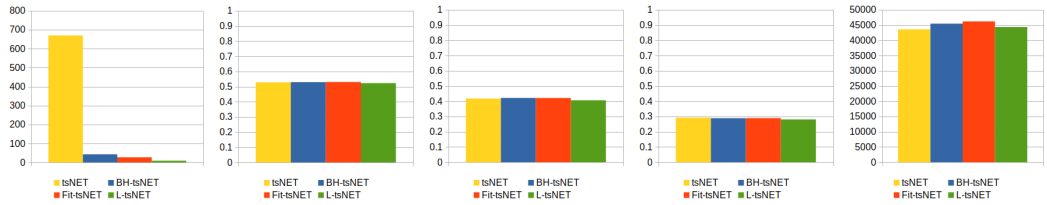
Table 1 Time complexity analysis of fast `tsNET` algorithms.

Algorithm	Shortest Path	KL Divergence	Entropy	Overall
`tsNET`	$O(nm)$	$O(n^2)$	$O(n^2)$	$O(nm)$
`BH-tsNET`	$\boldsymbol{O(n)}$	$O(n \log n)$	$\boldsymbol{O(n \log n)}$	$\boldsymbol{O(n \log n)}$
`FIt-tsNET`	$\boldsymbol{O(n)}$	$O(n)$	$\boldsymbol{O(n \log n)}$	$\boldsymbol{O(n \log n)}$
`L-tsNET`	$\boldsymbol{O(n)}$	$O(n)$	$\boldsymbol{O(n)}$	$\boldsymbol{O(n)}$

2 Comparison Experiments

We implement `BH-tsNET`, `FIt-tsNET`, and `L-tsNET` in Python using the `t-SNE` from the scikit-learn library [9], and in C++ based on `BH-SNE` [10] and `FIt-SNE` [5]. We use benchmark graphs used in the `tsNET` evaluation [3] as well as standard test suites commonly used in graph drawing studies: real-world scale-free graphs, biological networks, and mesh graphs. For the details, see the full version of this poster [7].

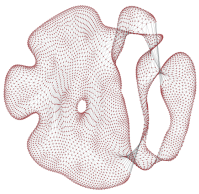
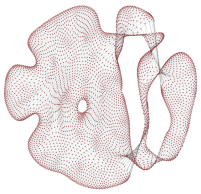
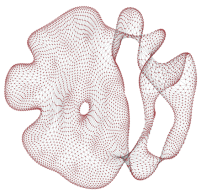
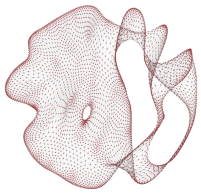
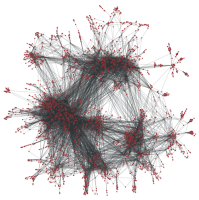
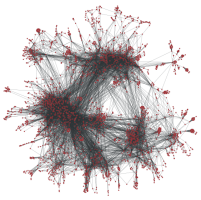
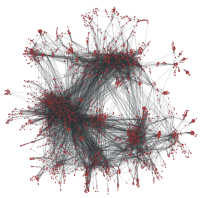
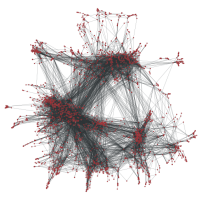
Extensive evaluation demonstrates that `BH-tsNET`, `FIt-tsNET`, and `L-tsNET` obtain significant runtime improvements over `tsNET`, on average 93.5%, 96%, and 98.6% respectively. See Figure 1(a).



(a) Runtime **(b)** Neigh. preserv. **(c)** Stress **(d)** Shape-based **(e)** Edge cross.

Figure 1 Average runtime and quality metrics: `BH-tsNET`, `FIt-tsNET`, and `L-tsNET` all run significantly faster than `tsNET`, with highly similar quality metrics.

Table 2 `BH-tsNET`, `FIt-tsNET` and `L-tsNET` compute the same quality drawings as `tsNET`.

tsNET	BH-tsNET	FIt-tsNET	L-tsNET
3elt			
			
oflights			
			
GION_7			
			

Surprisingly, our algorithms compute almost the same quality drawings as `tsNET`, in terms of quality metrics (neighborhood preservation [3], stress [1], edge crossings, shape-based metrics [2]), see Figures 1(b)-(e). Visual comparisons also confirm that our algorithms produce almost the same drawings as `tsNET`, see Table 2.

Moreover, we compare our algorithms to `DRGraph`, another linear-time `t-SNE`-based graph drawing algorithm with a different optimization function. Experiments show that our algorithms perform especially well over `DRGraph` in visualizing graphs with a regular structure, such as mesh graphs. For the details, see the full version of this poster [7].

Recently, we also presented fast GUMAP algorithms, designed explicitly for graph drawing based on UMAP, which reduce the time complexity from $O(nm)$ to $O(n^2 \log n)$ and $O(n)$, utilizing spectral sparsification and edge sampling. For the details, see [6].

References

1 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G Tollis. *Graph drawing: algorithms for the visualization of graphs*. Pearson, 1998.

2 Peter Eades, Seok-Hee Hong, An Nguyen, and Karsten Klein. Shape-based quality metrics for large graph visualization. *JGAA*, 21(1):29–53, 2017. doi:10.7155/jgaa.00405.

3 Johannes F Kruiger, Paulo E Rauber, Rafael Messias Martins, Andreas Kerren, Stephen Kobourov, and Alexandru C Telea. Graph layouts by t-SNE. In *CGF*, volume 36(3), pages 283–294, 2017. doi:10.1111/cgf.13187.

4 Solomon Kullback. *Information theory and statistics*. Dover Publications Inc., 1997.

5 George C Linderman, Manas Rachh, Jeremy G Hoskins, Stefan Steinerberger, and Yuval Kluger. Fast interpolation-based t-SNE for improved visualization of single-cell rna-seq data. *Nature methods*, 16(3):243–245, 2019.

6 Amyra Meidiana and Seok-Hee Hong. SS-GUMAP, SL-GUMAP, SSSL-GUMAP: fast UMAP algorithms for large graph drawing. *CoRR*, abs/2509.19703, 2025. doi:10.48550/ARXIV.2509.19703.

7 Amyra Meidiana, Seok-Hee Hong, and Kwan-Liu Ma. BH-tsNET, FIt-tsNET, L-tsNET: fast tsNET algorithms for large graph drawing. *CoRR*, abs/2509.19785, 2025. doi:10.48550/ARXIV.2509.19785.

8 Henri J Nussbaumer. The fast fourier transform. In *Fast Fourier Transform and Convolution Algorithms*, pages 80–111. Springer, 1981.

9 Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *JMLR*, 12:2825–2830, 2011. doi:10.5555/1953048.2078195.

10 Laurens Van Der Maaten. Accelerating t-SNE using tree-based algorithms. *JMLR*, 15(1):3221–3245, 2014. doi:10.5555/2627435.2697068.

11 Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *HMLR*, 9(11), 2008.

BH-tsNET, FIt-tsNET, L-tsNET: Fast tsNET Algorithms for Large Graph Drawing

Amyra Meidiana, Seok-Hee Hong, Kwan-Liu Ma
The University of Sydney and UC Davis

tsNET adapts the popular DR method t-SNE for graph drawing to compute high-quality drawings, preserving the neighborhood and clustering, however, O(nm) runtime results in poor scalability for large graphs. We present three fast tsNET algorithms, by introducing new fast methods high-dimensional probabilities computation and entropy computation with fast t-SNE for KL divergence computation

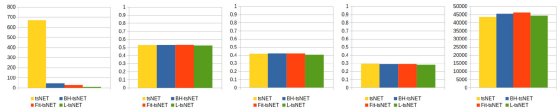
- **O(nlogn)-time BH-tsNET**: integrates a new partial BFS for all-pair shortest path computation, new quadtree-based entropy computation with fast BH t-SNE
- **O(nlogn)-time FIt-tsNET** : integrates a new partial BFS for all-pair shortest path computation, new quadtree-based entropy computation with fast Fit t-SNE
- **O(n)-time L-tsNET**: integrates a new partial BFS for all-pair shortest path computation, a new fast interpolation-based entropy computation with fast Fit t-SNE

Table 1 Time complexity analysis of fast tsNET algorithms

Algorithm	Shortest Path	KL Divergence	Entropy	Overall
tsNET	$O(nm)$	$O(n^2)$	$O(n^2)$	$O(nm)$
BH-tsNET	$\boldsymbol{O(n)}$	$O(n \log n)$	$\boldsymbol{O(n \log n)}$	$\boldsymbol{O(n \log n)}$
FIt-tsNET	$\boldsymbol{O(n)}$	$O(n)$	$\boldsymbol{O(n \log n)}$	$\boldsymbol{O(n \log n)}$
L-tsNET	$\boldsymbol{O(n)}$	$O(n)$	$\boldsymbol{O(n)}$	$\boldsymbol{O(n)}$

Extensive experiments using benchmark data sets confirm that

- **BH-tsNET, FIt-tsNET**, and **L-tsNET** outperform tsNET (**93.5%, 96%,** and **98.6%** faster runtime, respectively)
- preserving similar quality drawings in terms of quality metrics (**neighborhood preservation, stress, shape-based metrics**, and **edge crossing**)
- computing similar quality drawings by visual comparison



(a) Runtime **(b)** Neigh. preserv. **(c)** Stress **(d)** Shape-based **(e)** Edge cross.

Figure 1 Average runtime and quality metrics: BH-tsNET, FIt-tsNET, and L-tsNET all run significantly faster than tsNET, with very similar quality metrics.

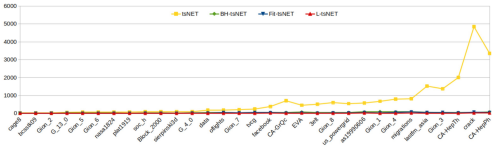


Fig. 2: Runtime (in seconds): On average, all of BH-tsNET, FIt-tsNET, and L-tsNET run 93%, 96%, and 98.6% faster than tsNET respectively.

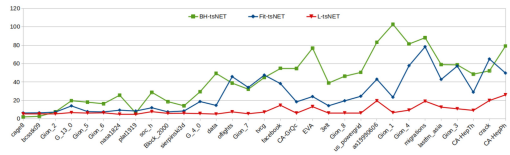


Fig. 3: Runtime comparison (in seconds) with tsNET redacted. L-tsNET run on average 78.6% and 66.3% faster than BH-tsNET and FIt-tsNET

Table 2 BH-tsNET, FIt-tsNET and L-tsNET compute the same quality drawings as tsNET.

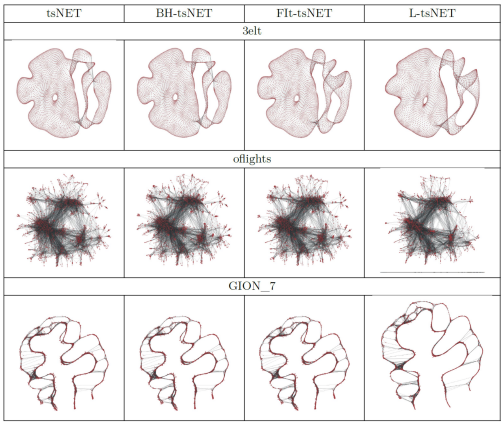


Table 6: Data sets for experiments

(a) Benchmark graphs

graph	$\|V\|$	$\|E\|$	u	density
G_13_0	1647	6487	300	3.94
soc_h	2000	16097	400	8.05
Block_2000	2000	9912	40	4.96
G_4_0	2075	4769	100	2.30
oflights	2905	15645	300	5.39
tveg	3213	10140	100	3.16
facebook	4039	88234	1100	21.8
CA-GrQc	4158	13422	40	3.23
EVA	4475	4652	600	1.04
us_powergrid	4941	6594	40	1.33
as19990606	5188	9930	1200	1.91
migrations	6025	9378	600	1.57
lastfm_asia	7624	27906	100	3.65
CA-HepTh	8638	24827	100	2.87
CA-HepPh	11204	117649	500	10.5

(b) GION graphs

graph	$\|V\|$	$\|E\|$	u	density
GION_2	1159	6424	40	5.54
GION_5	1748	13957	100	7.98
GION_6	1785	20459	100	11.5
GION_7	3010	41757	100	13.9
GION_8	4924	52502	100	10.7
GION_1	5452	118404	200	21.7
GION_4	5953	186279	200	31.3
GION_3	7885	427406	500	54.2

(c) Mesh graphs

graph	$\|V\|$	$\|E\|$	u	density
cage8	1015	4994	40	4.92
bcsstk09	1083	8677	40	8.01
nasa1824	1824	18692	40	10.2
plat1919	1919	15240	40	7.94
sierpinski3d	2050	6144	40	3.00
3elt	4720	13722	40	2.91
crack	10240	30380	40	2.97

Figure 2 GD 2025 Poster.

EnMRgy: Energy Network Analysis in Mixed Reality

Lucas Joos
University of Konstanz, Germany

Maximilian T. Fischer
University of Konstanz, Germany

Alexander Frings
University of Konstanz, Germany

Daniel A. Keim
University of Konstanz, Germany

Abstract

The shifting and ever-growing demand for energy, for instance, driven by transformations towards new technologies such as electric vehicles, heat pumps, battery storage, or rooftop solar, requires urban infrastructure to adapt. Upgrading legacy infrastructure, such as undersized electric cables, is costly, time-consuming, and disruptive, and therefore requires a holistic perspective and thorough urban planning that considers multi energy systems and co-located utilities. We present `EnMRgy`, a mixed-reality decision-support system that enables experts and decision-makers to explore a city's energy distribution networks, together with demand simulations and scenarios for infrastructure development. Within an immersive 3D city context, an energy network such as a power grid, modelled as a weighted graph, is visualised. Interactive functionalities allow users to adjust visual representations and compare scenarios across three different views. Our work enables evidence-based strategic planning for future-ready energy networks.

2012 ACM Subject Classification Human-centered computing → Graph drawings

Keywords and phrases Energy, Node-Link Diagrams, Immersive Analytics, Mixed Reality

Digital Object Identifier 10.4230/LIPIcs.GD.2025.55

Category Poster Abstract

Funding The authors gratefully acknowledge financial support by the Federal Ministry for Economic Affairs and Climate Action (BMWK, grant No. 03EI1048D) and by the City of Konstanz's Smart Green City program as part of the "Model Projects Smart Cities" funding program of the German Federal Ministry for Housing, Urban Development and Building (BMWSB, grant No. KfW 13622889).

1 Introduction

As heat pumps and electric vehicles replace combustion-based heating and transport on the move towards an all-electric society, electricity demand, especially peak loads, change. Rooftop solar adds variable generation, and batteries add new flexibility. These shifts strain legacy distribution networks built for predictable, one-way flows. Upgrading these networks is costly, slow, and disruptive, and must be coordinated with street layouts, rights of way, and co-located utilities over multi-year (often decade-long) timelines. Because actions in one energy sector (electricity, heating, fossil) affect the others, planning needs a whole-system view of *multi-energy systems* (MES) rather than individual analyses [3, 6].

Today, planning work is scattered across specialised domain-specific simulators, GIS tools, spreadsheets, and static reports. Engineers run power-flow and thermal-hydraulic models, planners weigh costs, permits, and construction logistics, and decision-makers juggle trade-offs under uncertainty. Comparing alternatives, such as layout changes, segmentation, or new


© Lucas Joos, Maximilian T. Fischer, Alexander Frings, and Daniel A. Keim;
licensed under Creative Commons License CC-BY 4.0





Figure 1 The EnMRgy system in the MR environment, consisting of the visual representation embedded in a 3D city model, a settings menu, and a slider to adjust the daytime used for demand simulation. Shown is a non-restricted version of an actual electricity distribution network.

supplies, often devolves into manual side-by-side checks. Time-varying loads, grid topology, and construction impacts (like work zones, road closures, cultural heritage protection) make it hard to see when and where bottlenecks arise. The complexity of this task requires comparative, context-rich, interpretable views [2]. Spatial optimisation links routing, cost, and geospatial constraints [9], but results are rarely explored interactively across disciplines, and cross-model comparisons remain opaque [1].

Mixed reality (MR), encompassing augmented reality (AR) and virtual reality (VR), can help address these challenges. Prior work shows that comparisons of weighted networks benefit from MR [5], especially when a three-dimensional context, such as a city model is of relevance [4]. Beyond stereoscopic vision, natural navigation can improve spatial understanding and make visual exploration more engaging [7] and enhanced visual comparison of differences between scenarios in situ. We present EnMRgy, an MR-based immersive analytics system that supports strategic MES planning for power grids.

2 The EnMRgy System

The EnMRgy (see Figure 1) provides an immersive MR environment, built with Unity and deployed on the Apple Vision Pro (2024). Interaction leverages built-in eye tracking and hand tracking for natural, controller-free gestures (e.g., gaze-and-pinch selection, manipulation, and navigation). Users can switch between AR, where surroundings and team members remain visible for collaboration, and VR for focused individual work. The system also allows users to switch between 2D ground surfaces, simplified 3D building shapes, or a photorealistic 3D model representing the city context. The weighted network, showing either cable capacities or utilisation from time-based demand simulations, is embedded in the city model (see Figure 2, top). Nodes represent consumers, junctions, and transformers. The tool leverages results from established power-flow and load-forecasting engines across energy sectors to visualise and compare critical system parameters. Simulations are typically run with specialised solvers or algorithms (e.g., PowerFactory, nPro, SPICE) using physical approximations and differential-algebraic equations (DAE). For multi-energy systems, mixed-integer optimisation

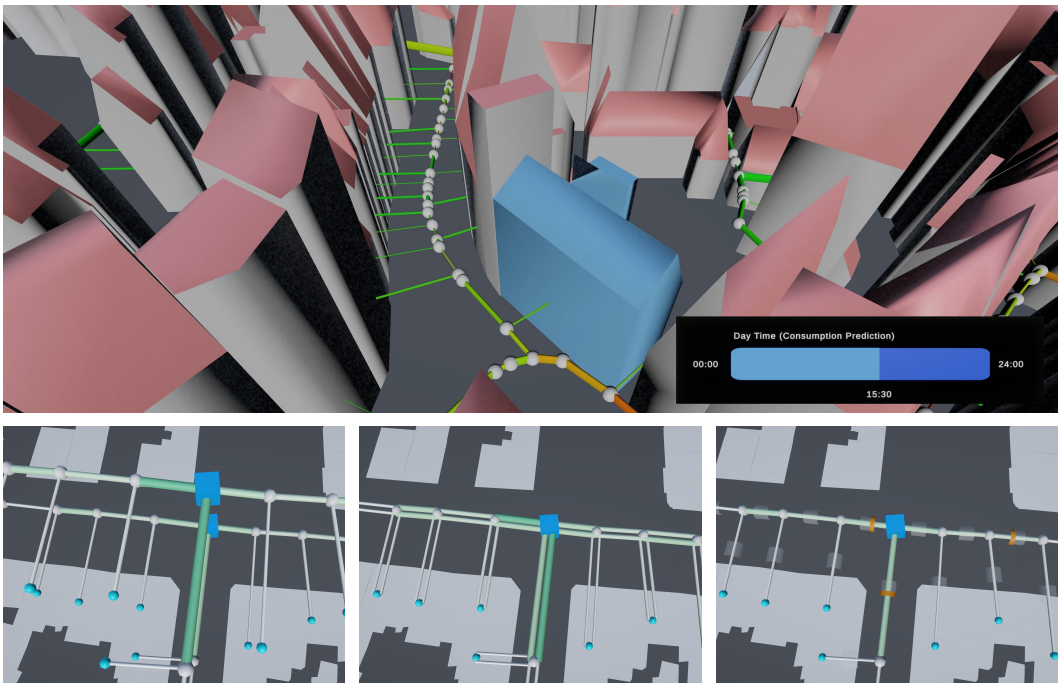


Figure 2 Besides the ground surfaces of buildings, `EnMRgy` can show the 3D shapes of buildings (top). Through super-positional, parallel, and glyph-based network comparison representations, users can compare the current network to scenarios where cables are replaced (bottom, left to right).

is thoroughly explored [8]. A time slider lets users set the time of day used for the simulation. Users can view a single network (single mode) or compare two networks (multi mode) via (i) super-positional alignment, (ii) parallel embedding, or (iii) glyph-based representation, where differences are shown as glyphs (e.g., a 20% capacity increase) (see Figure 2, bottom, left to right). Additional visual analytics features, such as details-on-demand and filtering, complement the tool.

3 Future Perspectives

We will broaden the system's scenario-based analysis and comparison capabilities. Planned extensions include (i) side-by-side comparison of networks with different topologies, (ii) richer demand simulations with more parameters, and (iii) additional integration of insights from recent VR research on network analysis. We will co-design additional features with domain experts to better support their workflows. Finally, we will evaluate the system's applicability and impact through a formal user study and expert feedback.

4 Conclusion

In this work, we present `EnMRgy`, a mixed-reality tool for supporting decisions targeting the development and upgrade of urban energy infrastructure. The graph-based visualisation of the power grid, embedded in the 3D city context enables users to understand the current state and constraints. Through scenario-based comparison techniques and demand simulation, decision-makers can explore alternatives, advance evidence-based strategic planning, and strive toward resilient and future-ready energy network systems.

References

1 Kaustav Bhattacharjee, Soumya Kundu, Indrasis Chakraborty, and Aritra Dasgupta. Who Should I Trust? A Visual Analytics Approach for Comparing Net Load Forecasting Models. In *2025 IEEE PES Grid Edge Technologies Conference & Exposition (Grid Edge)*, pages 1–5, 2025. doi:10.1109/GridEdge61154.2025.10887523.

2 Maximilian T. Fischer and Daniel A. Keim. Towards a Survey of Visualization Methods for Power Grids. *arXiv*, 2022. doi:10.48550/arXiv.2106.04661.

3 Wei Gu, Zhi Wu, Rui Bo, Wei Liu, Gan Zhou, Wu Chen, and Zaijun Wu. Modeling, planning and optimal energy management of combined cooling, heating and power microgrid: A review. *International Journal of Electrical Power & Energy Systems*, 54:26–37, 2014. doi:10.1016/j.ijepes.2013.06.028.

4 Lucas Joos, Maximilian T. Fischer, Julius Rauscher, Daniel A. Keim, Tim Dwyer, Falk Schreiber, and Karsten Klein. Visual network analysis in immersive environments: A survey, 2025. doi:10.48550/arXiv.2501.08500.

5 Lucas Joos, Sabrina Jaeger-Honz, Falk Schreiber, Daniel A. Keim, and Karsten Klein. Visual Comparison of Networks in VR. *IEEE Transactions on Visualization and Computer Graphics*, 28(11):3651–3661, 2022. doi:10.1109/TVCG.2022.3203001.

6 Giulia Mancò, Umberto Tesio, Elisa Guelpa, and Vittorio Verda. A review on multi energy systems modelling and optimization. *Applied Thermal Engineering*, 236:121871, 2024. doi:10.1016/j.applthermaleng.2023.121871.

7 Kim Marriott, Falk Schreiber, Tim Dwyer, Karsten Klein, Nathalie Henry Riche, Takayuki Itoh, Wolfgang Stuerzlinger, and Bruce H. Thomas. *Immersive analytics*, volume 11190. Springer, 2018. doi:10.1007/978-3-030-01388-2.

8 Salman Mashayekh, Michael Stadler, Gonçalo Cardoso, and Miguel Heleno. A mixed integer linear programming approach for optimal der portfolio, sizing, and placement in multi-energy microgrids. *Applied Energy*, 187:154–168, 2017. doi:10.1016/j.apenergy.2016.11.020.

9 Jun Shu, Lei Wu, Zuyi Li, Mohammad Shahidehpour, Lizi Zhang, and Bing Han. A New Method for Spatial Power Network Planning in Complicated Environments. *IEEE Transactions on Power Systems*, 27(1):381–389, 2012. doi:10.1109/TPWRS.2011.2161351.



Figure 3 The poster.

Using Reinforcement Learning to Optimize the Global and Local Crossing Number

Timo Brand
Technical University of Munich, Germany

Henry Förster
Technical University of Munich, Germany

Stephen Kobourov
Technical University of Munich, Germany

Robin Schukrafft
Technical University of Munich, Germany

Markus Wallinger
Technical University of Munich, Germany

Johannes Zink
Technical University of Munich, Germany

Abstract

We present a novel approach to graph drawing based on reinforcement learning for minimizing the global and the local crossing number, that is, the total number of edge crossings and the maximum number of crossings on any edge, respectively. An agent learns how to move a vertex based on a given observation vector. The agent receives feedback in the form of local reward signals tied to crossing reduction. To generate an initial layout, we use a stress-based graph-drawing algorithm. We compare our method against force- and stress-based baseline algorithms as well as three established algorithms for global crossing minimization on a suite of benchmark graphs. The experiments show mixed results: our current algorithm is mainly competitive for the local crossing number.

2012 ACM Subject Classification Theory of computation → Reinforcement learning; Theory of computation → Computational geometry

Keywords and phrases Reinforcement Learning, Crossing Minimization, Local Crossing Number

Digital Object Identifier 10.4230/LIPIcs.GD.2025.56

Category Poster Abstract

Related Version *Full Version*: `https://arxiv.org/abs/2509.06108`

1 Introduction

In *reinforcement learning* (RL), an agent receives a description of its environment, based on which it chooses an action changing the environment The agent receives a *reward*, which is positive for a "good" change or negative otherwise. The agent tries to maximize its reward. RL programs like Alpha(Go)Zero [8, 9] were able to master games like Go, chess, and Shogi by just being taught the rules and objectives of the game. They were not given any strategy. We investigate the following question: *Can we use reinforcement learning to optimize graph drawings with respect to a given graph drawing aesthetics measure?* We present an RL framework for graph drawings to optimize local and global crossings.[1] We perform a quantitative analysis on two graph data sets comparing our models against other methods.

[1] With *local crossings* we refer to the maximum number of crossings an edge has in a straight-line graph drawing. With *global crossings* we refer to the total number of crossings in a straight-line graph drawing.


© Timo Brand, Henry Förster, Stephen Kobourov, Robin Schukrafft, Markus Wallinger, and Johannes Zink;
licensed under Creative Commons License CC-BY 4.0



2 Description of the Approach

To obtain a size-invariant algorithm, we follow an idea of Safarli, Zhou, and Wang [7], who let an agent "sit" on a vertex and use as possible actions moving the vertex one step octagonally.

We provide the agent an observation vector describing the local neighborhood of a vertex v with respect to the eight octants defined by the eight octagonal rays around v. The first eight entries in the observation vector of v count the number of neighboring vertices in each octant, then eight entries count the total number of vertices. The next eight entries describe the distance to the closest neighbor, while the next eight entries describe the distance to the closest vertex. Finally, eight entries describe the amount of crossings that occur on edges incident to v in the respective octants and eight entries their local crossing number.

In each step, the agent has eight actions to choose from, which correspond to moving the vertex in one of the 16 main direction 22.5 degrees apart. The reward function is, for the global crossing number, the number of removed crossings. For the local crossing number, it is a combination of local and global crossing number. The choice of the vertex for the next step depends on a probability distribution weighted by crossings and vertex degree.

3 Experimental Evaluation

We compare our approach on the `rome` graphs [1] (relatively sparse; 10–100 vertices, 9–158 edges) and extended Barabási-Albert (`BA`) graphs [3] (random graphs; 50–150 vertices). An initial drawing used as input to the algorithms is obtained with the Kamada-Kawai algorithm [4]. We use 6,602 `rome` (1,000 `BA`) graphs for training and 1,650 (500) for testing. We record the global and local crossing number and the running time and use a time limit of 900 s; see Figure 1. The experiments ran on a server with a single CPU. The code is written in Python 3.12. The custom environment is implemented on top of the Gymnasium library, and we train an (artificial neural network) PPO agent using the stable-baselines3 library.

We compare our RL method trained for the global (RL(GC)) and local crossing number (RL(LC)) with a range of different methods: the *stress-minimization approach* Kamada-Kawai (KK) and the *spring embedder* Fruchterman-Reingold (FR), implemented in `networkx`, which do not directly optimize for (local) crossing number but generally achieve good results. We also compare with the machine-learning model SmartGD [10]; the authors released a model trained to minimize the crossing number, which we use. Regarding previous heuristic solutions, we compare to the *stochastic gradient descent* method $(\text{SGD})^2$ [2], to two *geometric local-optimization* approaches presented by Radermacher et al. [6], namely, Vertex Movement (VM) and Edge Insertion (EI), and to the *probabilistic hill-climbing* method Tübingen-Bus (TB). Note that while the latter approach was originally designed for minimizing the number of crossings in upward drawings, we could easily adjust it to drop the upward constraint [5].

4 Conclusion and Future Work

We have presented a post-processing procedure to optimize quantifiable objectives like the global and local crossing number. For the global crossing number, some state-of-the-art approaches achieve better results. This may depend on our current RL implementation. We made some ad-hoc design choices for the observation vector and the action space based on octants; also we chose a specific vertex-selection procedure and reward function. For the local crossing number, our algorithm is among the best-performing methods. Note, however, that we are among the first to explicitly optimize the local crossing number. In the future, we will try to apply more and better-suited RL approaches to graph drawing problems.

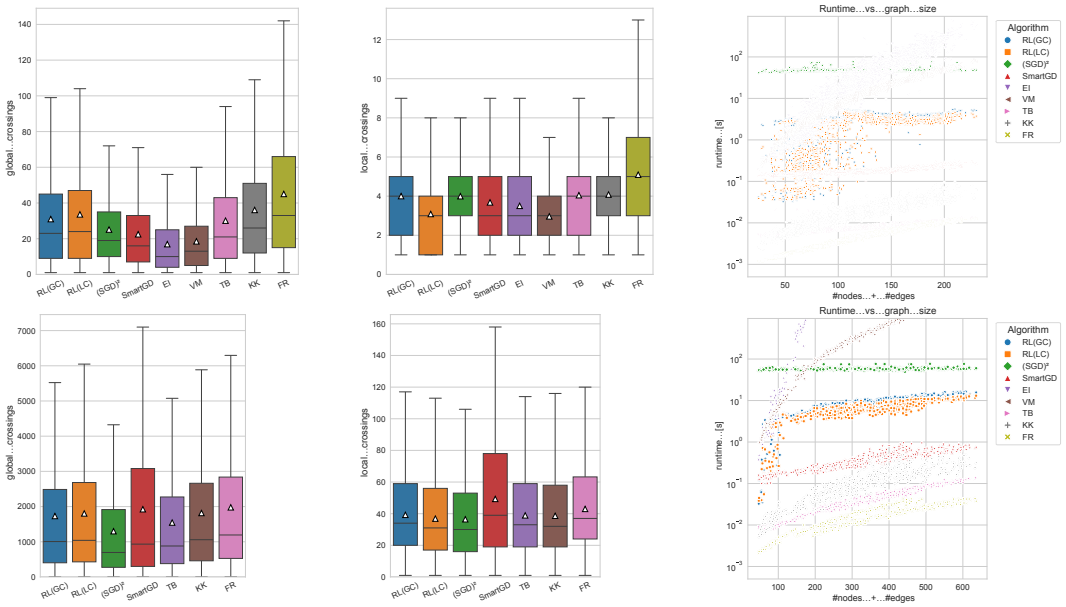


Figure 1 Boxplots for global (left) and local (middle) crossing number on the `rome` graphs (top) and `BA` graphs (bottom). On the right side, there are scatter plots for the running times.

References

1 Rome graphs. URL: `http://graphdrawing.org/data.html`.

2 Reyan Ahmed, Felice De Luca, Sabin Devkota, Stephen Kobourov, and Mingwei-ahmed2022multicriteria Li. Multicriteria scalable graph drawing via stochastic gradient descent, $(SGD)^2$. *IEEE Trans. Vis. Comput. Graph.*, 28(6):2388–2399, 2022. `doi:10.1109/TVCG.2022.3155564`.

3 Réka Albert and Albert-László Barabási. Topology of evolving networks: local events and universality. *Physical review letters*, 85(24):5234, 2000. `doi:10.1103/physrevlett.85.5234`.

4 Tomihisa Kamada and Satoru Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989. `doi:10.1016/0020-0190(89)90102-6`.

5 Maximilian Pfister. Personal communication, 2025.

6 Marcel Radermacher, Klara Reichard, Ignaz Rutter, and Dorothea Wagner. Geometric heuristics for rectilinear crossing minimization. *J. Exp. Algorithmics*, 24:1–21, 2019. `doi:10.1145/3325861`.

7 Ilkin Safarli, Youjia Zhou, and Bei Wang. Interpreting graph drawing with multi-agent reinforcement learning. *CoRR*, abs/2011.00748, 2020. `arXiv:2011.00748`.

8 David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018. `doi:10.1126/science.aar6404`.

9 David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017. `doi:10.1038/NATURE24270`.

10 Xiaoqi Wang, Kevin Yen, Yifan Hu, and Han-Wei Shen. SmartGD: A GAN-based graph drawing framework for diverse aesthetic goals. *IEEE Trans. Vis. Comput. Graph.*, 30(8):5666–5678, 2023. `doi:10.1109/TVCG.2023.3306356`.

A Poster

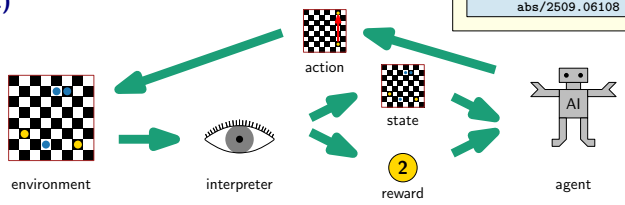
Using Reinforcement Learning to Optimize the Global and Local Crossing Number

Timo Brand, Henry Förster, Stephen Kobourov,
Robin Schukrafft, Markus Wallinger, and Johannes Zink

Full version: https://arxiv.org/abs/2509.06108

Reinforcement learning (RL)

- explore state space
- try to maximize reward
- learn from prior actions
- used in games like chess, go, etc. (e.g., AlphaGo, AlphaZero)
- used to tune large language models (LLMs) like ChatGPT
- powerful generic tool
- challenge: design states, rewards, possible actions, type of agent



Graph drawing as a game

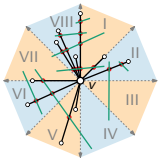
- choose measurable quality metric of a graph drawing like number of crossings or stress
- start with some initial drawing; quality is a number
- give the agent a state derived from the current drawing
- give the agent a set of possible actions
- for each choice of an action, update the graph drawing and re-compute the quality
- give the gain/loss of quality as reward to the agent

Experimental setup

- implemented in Python with the Gymnasium library
- PPO agent trained using stable-baselines3
- Trained on 6,602 `rome` and 1,000 `Barabasi-Albert` graphs, degree-1 vertices and planar graphs removed
- tested on one CPU with a time limit of 900 seconds
- methods do not start with random drawings but with a drawing computed by the Kamada–Kawai method
- 1,650 `rome` graphs and 500 `BA` graphs for testing

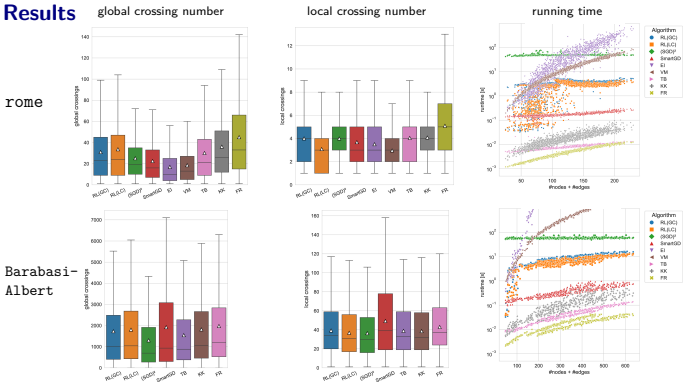
Minimizing the (rectilinear) local crossing number

- for size-invariance, we follow ideas of Safarli, Zhou, and Wang (2020)
- measurable quality metric: maximum number of crossings in which any edge is involved
- start with a random graph drawing
- iterate through the vertices
- give the agent the view of current vertex v:
  - state space: view of the 8 octants between the octagonal lines; for each octant number of neighbors, closest neighbor, closest non-neighbor, number of crossings of incident edges (all values normalized)
  - possible actions: move v along one of the 16 hexadecagonal lines
  - reward: decrease of the local crossing number plus (weighted less) decrease of the global crossing number; small negative reward if no progress is made



octant	I	II	III	IV	V	VI	VII	VIII
neighbors	.1	.2	0	0	.2	.1	0	.4
vertices	.14	.03	0	.11	.08	.2	.27	.17
closest ngb.	.32	.5	0	0	.25	.67	0	.21
closest vtx.	.32	.28	0	.94	.25	.67	.17	.21
crossings	.1	.2	0	0	.1	.3	0	1.0
local cr.	1	2	0	0	1	2	0	4

Results



- RL(GC): RL trained for global crossings
- RL(LC): RL trained for local crossings
- $(SGD)^2$: Stochastic Gradient Descent [1]
- SmartGD: GAN-based deep learning framework [2]
- EI: Edge Insertion algorithm from [3]
- VM: Vertex Movement algorithm from [3]
- TB: TübingenBus, probabilistic hill-climbing [4]
- KK: Kamada-Kawai method
- FR: Fruchterman-Reingold method

[1] R. Ahmed, F. De Luca, S. Devkota, S. Kobourov, and M. Li. Multicriteria scalable graph drawing via stochastic gradient descent, $(SGD)^2$. *IEEE Transactions on Visualization and Computer Graphics*, 2022.
[2] X. Wang, K. Yen, Y. Hu, and H.-W. Shen. SmartGD: A GAN-based graph drawing framework for diverse aesthetic goals. *IEEE Transactions on Visualization and Computer Graphics*, 2023.
[3] M. Radermacher, K. Reichard, I. Rutter, and D. Wagner. Geometric heuristics for rectilinear crossing minimization. *Journal of Experimental Algorithmics (JEA)*, 2019.
[4] Maximilian Pfister. Code for the Upward Crossing Minimization GD Contest 2019 and modified to drop the upward constraint. (Personal communication.)

Discussion

- Global crossing number: RL not among the best-performing approaches. This might be due to the design choices in our current RL implementation.
- Local crossing number: RL is among the best-performing methods. But, local crossing number has been hardly investigated and optimized. The other methods are generic or optimizing the global crossing number.
- Running time: large difference between algorithms and how they scale. RL is mid-table and can still handle the little larger `BA` graphs.

Conclusion and future work

- New post-processing procedure for graph drawing problems optimizing quantifiable objectives like global & local crossing number.
- Initial hope: plug in crossing minimization problem to some out-of-the-box RL solution. Turns out: not so obvious how, modeling seems to be crucial for learning success.
- Tune design of this RL approach. Apply more/different RL methods & frameworks to this and other graph drawing problems.