

Fabrizio Frati
Kwan-Liu Ma (Eds.)

LNCS 10692

Graph Drawing and Network Visualization

25th International Symposium, GD 2017
Boston, MA, USA, September 25–27, 2017
Revised Selected Papers



 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, Lancaster, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Zurich, Switzerland

John C. Mitchell

Stanford University, Stanford, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Dortmund, Germany

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbrücken, Germany

More information about this series at <http://www.springer.com/series/7407>

Fabrizio Frati · Kwan-Liu Ma (Eds.)

Graph Drawing and Network Visualization

25th International Symposium, GD 2017
Boston, MA, USA, September 25–27, 2017
Revised Selected Papers

Editors

Fabrizio Frati 
Roma Tre University
Rome
Italy

Kwan-Liu Ma 
University of California
Davis, CA
USA

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-73914-4 ISBN 978-3-319-73915-1 (eBook)
<https://doi.org/10.1007/978-3-319-73915-1>

Library of Congress Control Number: 2017963772

LNCS Sublibrary: SL1 – Theoretical Computer Science and General Issues

© Springer International Publishing AG 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This volume contains the papers presented at the 25th International Symposium on Graph Drawing and Network Visualization (GD 2017), which was held September 25–27, 2017, in Boston, Massachusetts, USA. 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. Information about the conference series and past symposia is maintained at <http://www.graphdrawing.org>. The 2017 edition of the conference was hosted by Northeastern University, with Cody Dunne and Alan Keahey as co-chairs of the Organizing Committee. The conference sessions took place in the Pavillion room of the Northeastern University Alumni Center at Columbus Place. A total of 85 participants attended the conference.

Regular papers could be submitted to one of two distinct tracks: Track 1 for papers on combinatorial and algorithmic aspects of graph drawing and Track 2 for papers on experimental, applied, and network visualization aspects. Short papers were reserved 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. In response to the call for papers, the Program Committee received a total of 107 submissions, consisting of 87 papers (35 in Track 1, 31 in Track 2, and 21 in the short paper category) as well as 20 posters. More than 320 expert reviews were provided, roughly a third of which were contributed by external reviewers. After extensive electronic discussions, the Program Committee selected 43 papers and 16 posters for inclusion in the scientific program of GD 2017. This resulted in an overall paper acceptance rate of 49% (65% in Track 1, 35% in Track 2, and 42% in the short paper category). This year it became mandatory for authors to publish an electronic version of their accepted papers on the ArXiv repository; a conference index with links to these contributions was made available before the conference.

There were two keynote talks at GD 2017. Timothy M. Chan, from the University of Illinois at Urbana-Champaign, USA, showed us how to have “Fun with Recursion and Tree Drawings.” Alessandro Vespignani, from the Northeastern University, USA, talked about methods for “Mapping the Next Pandemic.” Abstracts of both talks are included in the proceedings.

Springer sponsored awards for the best papers in Track 1 and Track 2, plus a best presentation award and a best poster award. As a result of a vote taken by the Program Committee, the award for the best paper in Track 1 was assigned to “Ordered Level Planarity, Geodesic Planarity, and Bi-Monotonicity” by Boris Klemz and Günter Rote, and the award for the best paper in Track 2 was assigned to “Revisited Experimental Comparison of Node-Link and Matrix Representations” by Mershack Okoe, Radu Jianu and Stephen Kobourov. The participants of the conference voted as the best presentation the one given by Philipp Kindermann for the paper “Experimental

Analysis of the Accessibility of Drawings with Few Segments” and as the best poster the one by Theresa Fröschl and Martin Nöllenburg entitled “Minimizing Wiggles in Storyline Visualizations.” Congratulations to all the award winners for their excellent contributions!

Following the tradition, the 24th Annual Graph Drawing Contest was held during the conference. The contest was divided into two parts – the creative topics and the live challenge – each with two categories, automatic and manual. The creative topics featured two graphs, one about citations among papers from previous GD symposia and one about human metabolism. The live challenge focused on maximizing the minimum crossing angle in straight-line drawings. Awards were given in each of the four categories. We thank the Contest Committee for preparing interesting and challenging contest problems. A report about the contest is included in the proceedings.

Many people and organizations contributed to the success of GD 2017. We would like to thank the Program Committee members 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. Thanks to all the authors who chose GD 2017 as the publication venue for their research. The Organizing Committee did a terrific job; a big thanks goes to Cody Dunne and Alan Keahey, who co-chaired the committee, as well as to the other local organizers and volunteers. GD 2017 thanks the “gold” sponsor Tom Sawyer Software, the “silver” sponsor yWorks, and the “bronze” sponsor Springer. Their generous support helps to ensure the continued success of this conference.

The 26th International Symposium on Graph Drawing and Network Visualization (GD 2018) will take place in September 2018 in Barcelona, Spain. Therese Biedl and Andreas Kerren will co-chair the Program Committee, Vera Sacristán and Rodrigo Silveira will co-chair the Organizing Committee.

November 2017

Fabrizio Frati
Kwan-Liu Ma

Organization

Steering Committee

Therese Biedl	University of Waterloo, Canada
Giuseppe Di Battista	Università Roma Tre, Italy
Fabrizio Frati	Università Roma Tre, Italy
Andreas Kerren	Linnaeus University, Sweden
Michael T. Goodrich	University of California at Irvine, USA
Yifan Hu	Yahoo Research, USA
Giuseppe Liotta (Chair)	Università di Perugia, Italy
Kwan-Liu Ma	University of California at Davis, USA
Martin Nöllenburg	Technische Universität Wien, Austria
Roberto Tamassia	Brown University, USA
Ioannis G. Tollis	FORTH-ICS and University of Crete, Greece
Alexander Wolff	Universität Würzburg, Germany

Program Committee

Daniel Archambault	Swansea University, UK
Benjamin Bach	University of Edinburgh, UK
Fabian Beck	Universität Duisburg-Essen, Germany
Michael Bekos	Universität Tübingen, Germany
Therese Biedl	University of Waterloo, Canada
Giordano Da Lozzo	University of California at Irvine, USA
Vida Dujmović	University of Ottawa, Canada
Stephane Durocher	University of Manitoba, Canada
Tim Dwyer	Monash University, Australia
Fabrizio Frati (Co-chair)	Università Roma Tre, Italy
Martin Gronemann	Universität zu Köln, Germany
John Alexis Guerra Gómez	Los Andes University, Colombia, and Berkeley, USA
Michael Hoffmann	ETH Zürich, Switzerland
Yifan Hu	Yahoo Research, USA
Takayuki Itoh	Ochanomizu University, Japan
Anna Lubiw	University of Waterloo, Canada
Kwan-Liu Ma (Co-chair)	University of California at Davis, USA
Fabrizio Montecchiani	Università di Perugia, Italy
Martin Nöllenburg	Technische Universität Wien, Austria
Arnaud Sallaberry	LIRMM, France
Andrew Suk	University of Illinois at Chicago, USA
Antonios Symvonis	National Technical University of Athens, Greece
Ioannis Tollis	FORTH-ICS and University of Crete, Greece
Csaba Tóth	California State University Northridge, USA

Alexander Wolff Universität Würzburg, Germany
Jian Zhao FX Palo Alto Lab, USA

Organizing Committee

Cody Dunne Northeastern University, USA
Alan Keahey Conversant, USA

Contest Committee

Philipp Kindermann FernUniversität in Hagen, Germany
Maarten Löffler (Chair) Utrecht University, The Netherlands
Ignaz Rutter Technische Universiteit Eindhoven, The Netherlands
Will Devanny University of California at Irvine, USA

Additional Reviewers

Shivam Agarwal	Luca Grilli	Saeed Mehrabi
Marco Angelini	Karen Gunderson	Debajyoti Mondal
Patrizio Angelini	Daniel Harabor	Pat Morin
Alessio Arleo	Haleh Havvaei	Lev Nachmanson
Alan Arroyo	Jose Tiberio Hernandez	Cydney Nielsen
David Auber	Marcel Hlawatsch	Dömötör Pálvölgyi
Yeganeh Bahoo	Christophe Hurter	Maurizio Patrignani
Juan Jose Besa Vial	Juan C. Ibarra Lopez	Michael Pelsmajer
Carla Binucci	Dino Ienco	Claire Pennarun
Johannes Blum	Veronika Irvine	Emmanuel Pietriga
Prosenjit Bose	Timothy Johnson	Alexander Pilz
Romain Bourqui	Sanjay Kairam	Sergey Pupyrev
Guido Brückner	Konstantinos Kakoulis	Chrysanthi Raftopoulou
Hsien-Chih Chang	Michael Kaufmann	Alexander Ravsky
Steven Chaplick	Philipp Kindermann	Vincenzo Roselli
Maxime Cordeil	Karsten Klein	Natan Rubin
Anthony D'Angelo	Stephen Kobourov	Ignaz Rutter
Éric Colin de Verdière	Myroslav Kryven	Christiane Spisla
William E. Devanny	Shahid Latif	Sabine Storandt
Emilio Di Giacomo	Fritz Lekschas	Alessandra Tappini
Walter Didimo	Fabian Lipp	Marc Van Kreveld
Thomas C. Van Dijk	Zhicheng Liu	Sue Whitesides
William Evans	Andre Löffler	Tilo Wiedera
Krzysztof Fleszar	Maarten Löffler	Steve Wismath
Siwei Fu	Min Lu	Yanhong Wu
Radoslav Fulek	Sven Mallach	Michael Wybrow
Ellen Gethner	Kim Marriott	Yalong Yang
Daniel Gonçalves	Fintan Mcgee	Wang Yong
Martin Graham	Tamara Mchedlidze	Jiawei Zhang

Sponsors

Gold Sponsor



Tom Sawyer[®]
SOFTWARE

Silver Sponsor



Bronze Sponsor



Keynote Presentations

Fun with Recursion and Tree Drawings

Timothy M. Chan

Department of Computer Science, University of Illinois at Urbana-Champaign,
USA

tmc@illinois.edu

Abstract. Divide-and-conquer has always been one of my favorite algorithm design paradigms. In this talk, I will survey existing techniques on drawings of trees on grids with small area or width, which nicely illustrate the power of recursive thinking. I will also mention some new improved upper bounds (work in progress) for certain types of tree drawings. Along the way, we will encounter a number of interesting functions and recurrences, and plenty of open problems.

Mapping the Next Pandemic

Alessandro Vespignani

College of Computer and Information Science, Northeastern University, USA
a.vespignani@northeastern.edu

Abstract. In the last ten years, we have seen dramatic advances in data collection and availability in a number of areas ranging from pathogen genetic sequences to human mobility patterns, and social media data. These advances, often dubbed as the “big data” revolution, have finally lifted many of the limitations affecting epidemic predictive modeling.

The results of these modeling approaches however must be communicated to policy makers and public health practitioners, possibly lifting the veil of the mathematical and statistical jargon. For this reason a visual approach to the data exploration/exploitation is often required. Here I will introduce recent development in the field and show some of the challenges to the development of visualization tools that show commonalities and patterns in emerging health threats, as well as explore the wide range of possible scenarios that can be used by policy makers to anticipate trends, evaluate risks, and eventually manage future pandemics.

Contents

Straight-Line Representations

Aligned Drawings of Planar Graphs	3
<i>Tamara Mchedlidze, Marcel Rademacher, and Ignaz Rutter</i>	
On the Edge-Length Ratio of Outerplanar Graphs	17
<i>Sylvain Lazard, William Lenhart, and Giuseppe Liotta</i>	
On Vertex- and Empty-Ply Proximity Drawings	24
<i>Patrizio Angelini, Steven Chaplick, Felice De Luca, Jiří Fiala, Jaroslav Hančl Jr., Niklas Heinsohn, Michael Kaufmann, Stephen Kobourov, Jan Kratochvíl, and Pavel Valtr</i>	
An Interactive Tool to Explore and Improve the Ply Number of Drawings . . .	38
<i>Niklas Heinsohn and Michael Kaufmann</i>	
Experimental Analysis of the Accessibility of Drawings with Few Segments	52
<i>Philipp Kindermann, Wouter Meulemans, and André Schulz</i>	

Obstacles and Visibility

Obstacle Numbers of Planar Graphs	67
<i>John Gimbel, Patrice Ossona de Mendez, and Pavel Valtr</i>	
Grid-Obstacle Representations with Connections to Staircase Guarding	81
<i>Therese Biedl and Saeed Mehrabi</i>	
Reconstructing Generalized Staircase Polygons with Uniform Step Length . . .	88
<i>Nodari Sitchinava and Darren Strash</i>	
3D Visibility Representations of 1-planar Graphs	102
<i>Patrizio Angelini, Michael A. Bekos, Michael Kaufmann, and Fabrizio Montecchiani</i>	

Topological Graph Theory

Lombardi Drawings of Knots and Links	113
<i>Philipp Kindermann, Stephen Kobourov, Maarten Löffler, Martin Nöllenburg, André Schulz, and Birgit Vogtenhuber</i>	

Arrangements of Pseudocircles: Triangles and Drawings	127
<i>Stefan Felsner and Manfred Scheucher</i>	
Drawing Bobbin Lace Graphs, or, Fundamental Cycles for a Subclass of Periodic Graphs	140
<i>Therese Biedl and Veronika Irvine</i>	
Many Touchings Force Many Crossings	153
<i>János Pach and Géza Tóth</i>	
Thrackles: An Improved Upper Bound.	160
<i>Radoslav Fulek and János Pach</i>	
Orthogonal Representations and Book Embeddings	
On Smooth Orthogonal and Octilinear Drawings: Relations, Complexity and Kandinsky Drawings	169
<i>Michael A. Bekos, Henry Förster, and Michael Kaufmann</i>	
EPG-representations with Small Grid-Size	184
<i>Therese Biedl, Martin Derka, Vida Dujmović, and Pat Morin</i>	
Mixed Linear Layouts of Planar Graphs.	197
<i>Sergey Pupyrev</i>	
Upward Partitioned Book Embeddings.	210
<i>Hugo A. Akitaya, Erik D. Demaine, Adam Hesterberg, and Quanquan C. Liu</i>	
Experimental Evaluation of Book Drawing Algorithms	224
<i>Jonathan Klawitter, Tamara Mchedlidze, and Martin Nöllenburg</i>	
Evaluations	
Visual Similarity Perception of Directed Acyclic Graphs: A Study on Influencing Factors	241
<i>K. Ballweg, M. Pohl, G. Wallner, and T. von Landesberger</i>	
GiViP: A Visual Profiler for Distributed Graph Processing Systems	256
<i>Alessio Arleo, Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani</i>	
Drawing Big Graphs Using Spectral Sparsification	272
<i>Peter Eades, Quan Nguyen, and Seok-Hee Hong</i>	
Revisited Experimental Comparison of Node-Link and Matrix Representations	287
<i>Mershack Okoe, Radu Jianu, and Stephen Kobourov</i>	

Tree Drawings

Improved Bounds for Drawing Trees on Fixed Points
with L-Shaped Edges. 305
*Therese Biedl, Timothy M. Chan, Martin Derka, Kshitij Jain,
and Anna Lubiw*

On Upward Drawings of Trees on a Given Grid. 318
Therese Biedl and Debajyoti Mondal

Simple Compact Monotone Tree Drawings. 326
Anargyros Oikonomou and Antonios Symvonis

Visualizing Co-phylogenetic Reconciliations. 334
*Tiziana Calamoneri, Valentino Di Donato, Diego Mariottini,
and Maurizio Patrignani*

Graph Layout Designs

Anisotropic Radial Layout for Visualizing Centrality
and Structure in Graphs. 351
Mukund Raj and Ross T. Whitaker

Computing Storyline Visualizations with Few Block Crossings. 365
*Thomas C. van Dijk, Fabian Lipp, Peter Markfelder,
and Alexander Wolff*

MLSEB: Edge Bundling Using Moving Least Squares Approximation. 379
Jieting Wu, Jianping Zeng, Feiyu Zhu, and Hongfeng Yu

Drawing Dynamic Graphs Without Timeslices 394
Paolo Simonetto, Daniel Archambault, and Stephen Kobourov

Point-Set Embeddings

Colored Point-Set Embeddings of Acyclic Graphs. 413
*Emilio Di Giacomo, Leszek Gasieniec, Giuseppe Liotta,
and Alfredo Navarra*

Planar Drawings of Fixed-Mobile Bigraphs 426
*Michael A. Bekos, Felice De Luca, Walter Didimo, Tamara Mchedlidze,
Martin Nöllenburg, Antonios Symvonis, and Ioannis G. Tollis*

Ordered Level Planarity, Geodesic Planarity and Bi-Monotonicity. 440
Boris Klemz and Günter Rote

Non-crossing Paths with Geographic Constraints. 454
Rodrigo I. Silveira, Bettina Speckmann, and Kevin Verbeek

Special Representations

Planar L-Drawings of Directed Graphs.	465
<i>Steven Chaplick, Markus Chimani, Sabine Cornelsen, Giordano Da Lozzo, Martin Nöllenburg, Maurizio Patrignani, Ioannis G. Tollis, and Alexander Wolff</i>	
NodeTrix Planarity Testing with Small Clusters	479
<i>Emilio Di Giacomo, Giuseppe Liotta, Maurizio Patrignani, and Alessandra Tappini</i>	
The Painter's Problem: Covering a Grid with Colored Connected Polygons	492
<i>Arthur van Goethem, Irina Kostitsyna, Marc van Kreveld, Wouter Meulemans, Max Sondag, and Jules Wulms</i>	
Triangle-Free Penny Graphs: Degeneracy, Choosability, and Edge Count. . . .	506
<i>David Eppstein</i>	

Beyond Planarity

1-Fan-Bundle-Planar Drawings of Graphs.	517
<i>Patrizio Angelini, Michael A. Bekos, Michael Kaufmann, Philipp Kindermann, and Thomas Schneck</i>	
Gap-Planar Graphs	531
<i>Sang Won Bae, Jean-Francois Baffier, Jinhee Chun, Peter Eades, Kord Eickmeyer, Luca Grilli, Seok-Hee Hong, Matias Korman, Fabrizio Montecchiani, Ignaz Rutter, and Csaba D. Tóth</i>	
Beyond Outerplanarity.	546
<i>Steven Chaplick, Myroslav Kryven, Giuseppe Liotta, Andre Löffler, and Alexander Wolff</i>	
The Effect of Planarization on Width.	560
<i>David Eppstein</i>	

Contest Report

Graph Drawing Contest Report.	575
<i>William Devanny, Philipp Kindermann, Maarten Löffler, and Ignaz Rutter</i>	

Poster Abstracts

Minimizing Wiggles in Storyline Visualizations 585
Theresa Fröschl and Martin Nöllenburg

Graph Drawing for Formalized Diagrammatic Proofs in Geometry 588
Nathaniel Miller

Drawing Graphs on Few Circles and Few Spheres 591
Myroslav Kryven, Alexander Ravsky, and Alexander Wolff

Counterexample to the Variant of the Hanani–Tutte Theorem
on the Genus-4 Surface 594
Radoslav Fulek and Jan Kynčl

A Geometric Heuristic for Rectilinear Crossing Minimization 597
*Marcel Radermacher, Klara Reichard, Ignaz Rutter,
and Dorothea Wagner*

A Note on Plus-Contacts, Rectangular Duals,
and Box-Orthogonal Drawings 600
Therese Biedl and Debajyoti Mondal

Grid Obstacle Representation of Graphs 603
*Arijit Bishnu, Arijit Ghosh, Rogers Mathew, Gopinath Mishra,
and Subhabrata Paul*

Summarizing and Visualizing Graph Ensembles with Rank
Statistics and Boxplots 606
Mukund Raj, Ian Ruginiski, Robert M. Kirby, and Ross T. Whitaker

Planar k -NodeTrix Graphs: A New Family of Beyond Planar Graphs 609
*Emilio Di Giacomo, Giuseppe Liotta, Maurizio Patrignani,
and Alessandra Tappini*

Towards Characterizing Strict Outerconfluent Graphs 612
Fabian Klute and Martin Nöllenburg

Flattening Polygonal Linkages via Uniform Angular Motion 615
*Hugo A. Akitaya, Matthew D. Jones, Gregory A. Sandoval,
Diane L. Souvaine, David Stalfá, and Csaba D. Tóth*

Optimal Compaction of Orthogonal Grid Drawings for Graphs
of Arbitrary Vertex Degrees 618
Eduardo Santiago Ramos and Adriano Chaves Lisboa

BCSA: BC Tree-Based Sampling and Visualization of Big Graphs 621
Seok-Hee Hong, Quan Nguyen, Amyra Meidiana, and Jiayi Li

Low Ply Drawings of Trees of Bounded Degree 624
Michael T. Goodrich and Timothy Johnson

Which Graph Layout Gives a Good Shape for Large Graphs? 627
Quan Nguyen, Peter Eades, and Seok-Hee Hong

MetagenomeScope: Web-Based Hierarchical Visualization
of Metagenome Assembly Graphs 630
Marcus Fedarko, Jay Ghurye, Todd Treangen, and Mihai Pop

Author Index 633

Straight-Line Representations

Aligned Drawings of Planar Graphs

Tamara Mchedlidze¹, Marcel Radermacher^{1(✉)}, and Ignaz Rutter²

¹ Department of Computer Science, Karlsruhe Institute of Technology,
Karlsruhe, Germany

`mched@iti.uka.de`, `radermacher@kit.edu`

² Department of Mathematics and Computer Science, TU Eindhoven,
Eindhoven, The Netherlands

`i.rutter@tue.nl`

Abstract. Let G be a graph embedded in the plane and let \mathcal{A} be an arrangement of pseudolines intersecting the drawing of G . An *aligned* drawing of G and \mathcal{A} is a planar polyline drawing Γ of G with an arrangement A of lines so that Γ and A are homeomorphic to G and \mathcal{A} . We show that if \mathcal{A} is stretchable and every edge e either entirely lies on a pseudoline or intersects at most one pseudoline, then G and \mathcal{A} have a straight-line aligned drawing. In order to prove these results, we strengthen the result of Da Lozzo et al. [5], and prove that a planar graph G and a single pseudoline \mathcal{L} have an aligned drawing with a prescribed convex drawing of the outer face. We also study the more general version of the problem where only a set of vertices is given and we need to determine whether they can be collinear. We show that the problem is \mathcal{NP} -hard but fixed-parameter tractable.

1 Introduction

Two fundamental primitives for highlighting structural properties of a graph in a drawing are alignment of vertices such that they are collinear and geometrically separating unrelated graph parts, e.g., separating them by a straight line. Not surprisingly, both these techniques have been previously considered from a theoretical point of view in the case of planar straight-line drawings.

Da Lozzo et al. [5] study the problem of producing a planar straight-line drawing of a given embedded graph $G = (V, E)$, i.e., G has a fixed combinatorial embedding and a fixed outer face, such that a given set $S \subseteq V$ of vertices is collinear. It is clear that if such a drawing exists, then the line containing the vertices in S is a simple curve starting and ending at infinity that for each edge e of G either fully contains e or intersects e in at most one point, which may be an endpoint. We call such a curve a *pseudoline* with respect to G . Da Lozzo et al. [5] show that this is a full characterization of the alignment problem, i.e., a straight-line drawing where the vertices in S are collinear exists if and

Work was partially supported by grant WA 654/21-1 of the German Research Foundation (DFG).

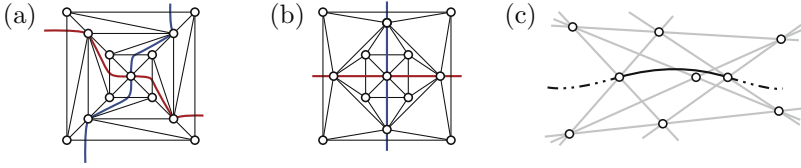


Fig. 1. Aligned Drawing (b) of a 2-aligned planar graph (a). The pseudolines \mathcal{R} and \mathcal{B} and the corresponding lines in the drawing are drawn red and blue, respectively. (c) A non-stretchable arrangement of 9 pseudolines, which can be seen as a stretchable arrangement of 8 pseudolines (grey) and an edge (black solid).

only if there exists a pseudoline \mathcal{L} with respect to G that contains the vertices in S . However testing whether such a pseudoline exists is an open problem, which we consider in this paper.

Likewise, for the problem of separation, Biedl et al. [1] considered so-called *HH*-drawings, where given an embedded graph $G = (V, E)$ and a partition $V = A \dot{\cup} B$, one seeks a planar drawing of G in which A and B can be separated by a line. Again, it turns out that such a drawing exists if and only if there exists a pseudoline \mathcal{L} with respect to G such that the vertices in A and B are separated by \mathcal{L} in the sense that they are in different *halfplanes*. Cano et al. [2] extend the result to planar straight-line drawings with a given star-shaped outer face.

In particular, the results of Da Lozzo et al. [5] show that given a pseudoline \mathcal{L} with respect to G one can always find a planar straight-line drawing of G such that the vertices on \mathcal{L} are collinear and the vertices contained in the halfplanes defined by \mathcal{L} are separated by a line L . In other words, a topological configuration consisting of a planar graph G and a pseudoline with respect to G can always be stretched. In this paper we initiate the study of this stretchability problem with more than one given pseudoline.

More formally, a pair (G, \mathcal{A}) is a *k-aligned graph* if $G = (V, E)$ is a planar embedded graph and $\mathcal{A} = \{\mathcal{L}_1, \dots, \mathcal{L}_k\}$ is an arrangement of (pairwise intersecting) pseudolines with respect to G . If the number k of curves is clear from the context, we drop it from the notation and simply speak of *aligned graphs*. For 1-aligned graphs we write (G, \mathcal{L}) instead of $(G, \{\mathcal{L}\})$. Let $A = \{L_1, \dots, L_k\}$ be a line arrangement and Γ be planar drawing of G . A tuple (Γ, A) is an *aligned drawing of (G, \mathcal{A})* if and only if the following properties hold; refer to Fig. 1(a-b). (i) The arrangement of A is homeomorphic to the arrangement of \mathcal{A} (i.e., \mathcal{A} is *stretchable*), (ii) Γ is homeomorphic to the planar embedding of G , (iii) each line L_i intersects in Γ the same vertices and edges as \mathcal{L}_i in G , and it does so in the same order. We focus on straight-line aligned drawings. For brevity, unless stated otherwise, the term aligned drawing refers to a straight-line drawing throughout this paper.

Note that the stretchability of \mathcal{A} is a necessary condition for the existence of an aligned drawing. Since testing stretchability is \mathcal{NP} -hard [14, 15], we assume that a geometric realization A of \mathcal{A} is provided. However, line arrangements of size up to 8 are always stretchable [10] and only starting from 9 lines

non-stretchable arrangements exist; see the Pappus configuration [11] in Fig. 1(c). It is conceivable that in practical applications, e.g., stemming from user interactions, the number of lines to stretch is small. The same configuration illustrates an example of an 8-aligned graph with a single edge that does not have an aligned drawing.

The aligned drawing convention generalizes the problems studied by Da Lozzo et al. and Biedl et al. who concentrated on the case of a single line. We study a natural extension of their setting and ask for alignment on general line arrangements.

In addition to the strongly related work mentioned above, there are several other works that are related to the alignment of vertices in drawings. Dujmović [6] shows that every n -vertex planar graph $G = (V, E)$ has a planar straight-line drawing such that $\Omega(\sqrt{n})$ vertices are aligned, and Da Lozzo et al. [5] show that in planar treewidth-3 graphs, one can align $\Theta(n)$ vertices and that in treewidth- k graphs one can align $\Omega(k^2)$ vertices. Chaplik et al. [3] study the problem of drawing planar graphs such that all edges can be covered by k lines. They show that it is \mathcal{NP} -hard to decide whether such a drawing exists. Deciding whether there exists a drawing where all vertices lie on k lines is open [4]. Drawings of graphs on n lines where a mapping between the vertices and the lines is provided have been studied by Dujmović et al. [7, 8].

Contribution & Outline. First we study the topological setting where we are given a planar graph G and set S of vertices to align in Sect. 3. We show that the problem is \mathcal{NP} -hard but fixed-parameter tractable (FPT) with respect to $|S|$. Afterwards in Sect. 4 we consider the geometric setting where we seek an aligned drawing of an aligned graph. In Sect. 4.2, we strengthen the result of Da Lozzo et al. and Biedl et al. and show that there exists a 1-aligned drawing of G with a given convex drawing of the outer face. In Sect. 4.3 we consider k -aligned graphs with a stretchable pseudoline arrangement, where every edge e either entirely lies on a pseudoline or intersects with at most one pseudoline, which can either be in the interior or an endpoint of e . We utilize the previous result to prove that every such k -aligned graph has an aligned drawing, for any value of k . The proofs of statements marked with (\star) can be found in the full version on arXiv [12].

2 Preliminaries

Let \mathcal{A} be a pseudoline arrangement of a set of k pseudolines $\mathcal{L}_1, \dots, \mathcal{L}_k$ and (G, \mathcal{A}) be an aligned graph. The set of cells in \mathcal{A} is denoted by $\text{cells}(\mathcal{A})$. A cell is *empty* if it does not contain a vertex of G . Removing from a pseudoline its intersections with other pseudolines gives a set of its *pseudosegments*.

Let $G = (V, E)$ be a planar embedded graph with a vertex set V and an edge set E . We call $v \in V$ *interior* if v does not lie on the boundary of the outer face of G . An edge $e \in E$ is *interior* if e does not lie entirely on the boundary of the outer face of G . An interior edge is a *chord* if it connects two vertices on the outer face. A point p of an edge e is an *interior point* of e if p is not an

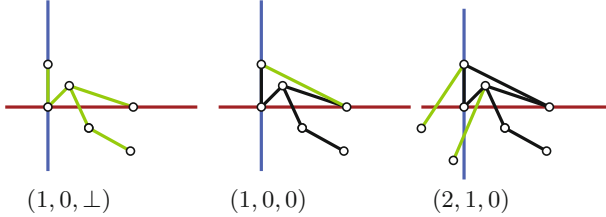


Fig. 2. Examples for the alignment complexity of an aligned graph.

endpoint of e . A *triangulation* is a planar embedded graph whose inner faces are all triangles and whose outer face is bounded by a simple cycle. A *triangulation* of a graph G is a triangulation that contains G as a subgraph. For a graph G and an edge e of G , not being an edge of a separating triangle, the graph G/e is obtained from G by contracting e and merging the resulting multiple edges and removing self-loops. A k -*wheel* is a wheel graph W_k with k vertices on the outer face and one interior vertex. Let Γ be a drawing of G and let C be a cycle in G . We denote with $\Gamma[C]$ the drawing of C in Γ . A k -*aligned triangulation* of (G, \mathcal{A}) is a k -aligned graph (G_T, \mathcal{A}) with G_T being a triangulation of G .

A vertex is \mathcal{L}_i -*aligned* (or simply *aligned* to \mathcal{L}_i) if it lies on the pseudoline \mathcal{L}_i . A vertex that is not aligned is *free*. An edge e is \mathcal{L}_i -*aligned* (or simply *aligned*) if it completely lies on \mathcal{L}_i . Let E_{aligned} be the set of all aligned edges. An *intersection vertex* lies on the intersection of two pseudolines \mathcal{L}_i and \mathcal{L}_j . An edge is i -*anchored* ($i = 0, 1, 2$) if i of its endpoints are aligned to distinct curves. Let E_i be the set of i -anchored edges; note that, the set of edges is the disjoint union $E_0 \cup E_1 \cup E_2$. A 0-anchored, 0-crossed, non-aligned edge is also called *free*. An edge e is (at most) l -*crossed* if (at most) l distinct pseudolines intersect e in its interior. A non-empty edge set $A \subset E$ is l -crossed if l is the smallest number such that every edge in A is at most l -crossed.

The *alignment complexity* of an aligned graph \mathcal{G} in a way describes how “complex” the relationship between the graph G and the line arrangement $\mathcal{L}_1, \dots, \mathcal{L}_k$ is and formally is defined as a triple (l_0, l_1, l_2) , where $l_i, i = 0, 1, 2$, describes the “complexity of i -anchored edges”, i.e. it indicates that E_i is at most l_i -crossed or has to be empty, if $l_i = \perp$. For example, an aligned graph where every vertex is aligned and every edge has at most l interior intersections has the alignment complexity (\perp, \perp, l) . For further examples, see Fig. 2.

3 Complexity and Fixed-Parameter Tractability

In this section we deal with the topological setting where we are given a planar embedded graph $G = (V, E)$ and a subset $S \subset V$ to be collinear. According to Da Lozzo et al. [5], this is equivalent to the existence of a pseudoline $\mathcal{L}(S)$ with respect to G passing exactly through the vertices in S . We refer to this problem as *pseudoline existence problem*. Using techniques similar to Föbmeier

and Kaufmann [9], we can show that the pseudoline existence problem is \mathcal{NP} -hard; see full version [12]. In the following, we show that the pseudoline existence problem is FPT with respect to $|S|$. In the first step deal with biconnected planar embedded graphs only. Additionally to the set of vertices S we require the pseudoline to pass through a set of faces F . This trick allows us to combine two independent pseudolines of two biconnected components of a planar embedded graph. Thus, let additionally G be biconnected and F be a subset of faces of G . Notice that in case that the vertices of S are not independent, they can only form a *linear forest*, i.e., a set of paths, as otherwise there is no pseudoline through the vertices S with respect to G . Let G^* be the dual graph of G (see Fig. 3.a) and let $S' \subseteq S$ be the subset of vertices that form end-points of the paths induced by S . We denote by f a face of G and by f^* its dual vertex. An *extended* dual $G_e^*(S, F)$ is the graph obtained from G^* by the following steps. We omit the parameters (S, F) if they are clear from the context.

- Step 1:** For each path induced by S that contains exactly one edge, subdivide the edge by a vertex and add it to S ,
- Step 2:** Place the vertices of S into the corresponding faces of G^* and the edges induced by them (red vertices and edges in Fig. 3(a)).
- Step 3:** Connect the vertices of S' to all vertices of the dual face they lie in (red dashed edges in Fig. 3(a)).
- Step 4:** For each vertex $v \in S$ remove the edges dual to the primal edges incident to v .
- Step 5:** Remove all the dual vertices that have in degree zero or one; see Fig. 3(b).
- Step 6:** Replace each vertex f^* of G^* with a clique of the size equal the degree of f^* , each vertex v of this clique corresponds to an edge e^* incident to a vertex f^* , thus we call it a *clique vertex* corresponding to e^* and denote by $\text{cl}(f, e^*)$. For each edge of G_e^* that has survived, we connect the two corresponding to it clique vertices.
- Step 7:** Recall that F is a set of faces of G . Assume we would like our pseudoline to pass through the faces of F . To check for existence of such a pseudoline we further augment the graph G_e^* as follows. For each $f \in F$, additionally to the clique that is built on face vertices $\text{cl}(f, e_1^*) \dots \text{cl}(f, e_a^*)$, corresponding to the edges incident to the dual vertex f , we add a star with a new center vertex $\text{cent}(f)$ that has $\text{cl}(f, e_1^*) \dots \text{cl}(f, e_a^*)$ as leaves. Finally, we set $\text{cent}(F) = \{\text{cent}(f) \mid f \in F\}$.

Lemma 1 (\star). *Let $G = (V, E)$ be a biconnected planar embedded graph, let $S \subset V$ be a set of vertices that induce a linear forest and let F be a set of faces of G . There exist an aligned graph (G, \mathcal{L}) , where \mathcal{L} passes through all vertices of S and faces F if and only if there exists a simple cycle C in the extended dual $G_e^*(S, F)$ through the vertices of $S \cup \text{cent}(F)$.*

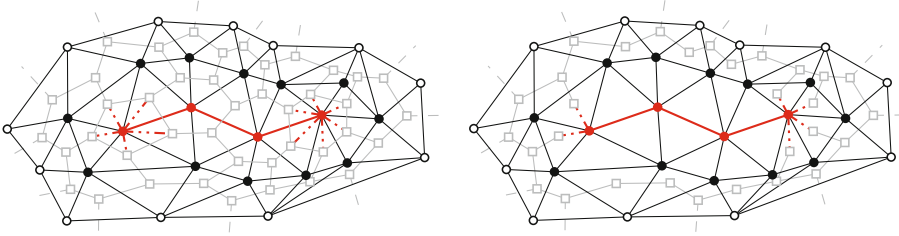


Fig. 3. A fragment of a graph G (black) and of $G_e^*(S, F)$ (gray and red) (a) after Step 3, (b) after Step 6.

We utilize the following theorem.

Theorem 1 (Wahlström [16]). *Given a graph $G = (V, E)$ and a subset $S \subset V$, it can be tested in $O(2^{|S|}\text{poly}(n))$ time whether a simple cycle through the vertices in S exists. In affirmative the cycle can be reported within the same asymptotic time.*

Theorem 1 together with Lemma 1 gives a $O(2^{|S|}\text{poly}(n))$ time algorithm to solve the pseudoline construction problem for the case of biconnected graphs. We next sketch how this can be extended to general planar graphs.

Lemma 2 (\star). *Let $G = (V, E)$ be a planar embedded graph, $S \subset V$ and c be a cut vertex of G separating G into subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Let $S_1 = S \cap V_1 \neq \emptyset$ and $S_2 = S \cap V_2 \setminus \{c\} \neq \emptyset$. Let f be the face of G_1 where G_2 lies. There exists an aligned graph $(G, \mathcal{L}(S))$ if and only if there exist aligned graphs $(G_1, \mathcal{L}(S_1))$ and $(G_2, \mathcal{L}(S_2))$, such that $\mathcal{L}(S_1)$ passes through face f .*

Utilizing Lemma 2, we can recursively decompose a graph into biconnected components, check for the pseudoline existence by applying Lemma 1 and Theorem 1 and glue the pseudolines if they exist. This implies the following:

Theorem 2 (\star). *Given a planar embedded graph $G = (V, E)$ and a subset $S \subset V$, it can be tested in $O(4^{|S|}\text{poly}(n))$ time whether an aligned graph $(G, \mathcal{L}(S))$ exists.*

4 Drawing Aligned Graphs

We show that every aligned graph where each edge either entirely lies on a pseudoline or is intersected by at most one pseudoline, i.e., alignment complexity $(1, 0, \perp)$, has an aligned drawing. For 1-aligned graphs we show the stronger statement that every 1-aligned graph has an aligned drawing with a given aligned convex drawing of the outer face. We first present our proof strategy and then deal with 1- and k -aligned graphs.

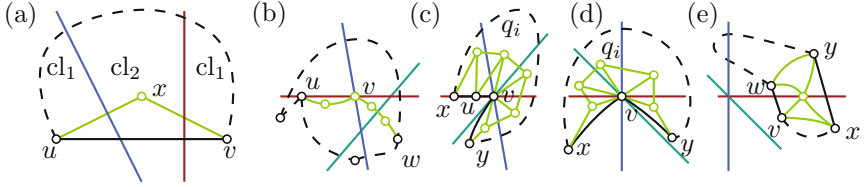


Fig. 4. Steps for triangulating aligned graphs (black) with 1-crossed edges (green).

4.1 Proof Strategy

Our general strategy for proving the existence of aligned drawings of an aligned graph (G, \mathcal{A}) is as follows. First, we show that we can triangulate (G, \mathcal{A}) by adding vertices and edges without invalidating its properties. We can thus assume that our aligned graph (G, \mathcal{A}) is an aligned triangulation. Second we show that, unless G is a specific graph (e.g., a k -wheel or a triangle), it contains a specific type of edge, namely an edge that is contained in a pseudoline, or an edge that is not intersected by any of the pseudolines. Third, we exploit the existence of such an edge to inductively prove the existence of an aligned drawing of (G, \mathcal{A}) . Depending on whether the edge is contained in a separating triangle or not, we either decompose along that triangle or contract the edge. In both cases the problem reduces to smaller instances that are almost independent. In order to combine solutions, it is, however, crucial to use the same arrangement of lines \mathcal{A} for both of them.

In the following we introduce the necessary tools used for all three steps on k -aligned graphs of alignment complexity $(1, 0, \perp)$. Recall, that for this class (i) every free edge is at most 1-crossed, (ii) every 1-anchored edge has no intersections, and (iii) there is no edge with its endpoints on two pseudolines. Lemma 3 shows that every aligned graph has an aligned triangulation with almost the same alignment complexity. If G contains a separating triangle, Lemma 4 shows that (G, \mathcal{A}) admits an aligned drawing if both split components have an aligned drawing. Finally, with Lemma 5 we obtain a drawing of (G, \mathcal{A}) from a drawing of the aligned graph $(G/e, \mathcal{A})$ where one special edge e is contracted. For simplicity we assume the input graph to be 2-connected, general graphs allow similar techniques.

Lemma 3 (\star). *Let (G, \mathcal{A}) be a biconnected k -aligned graph of alignment complexity $(2, 0, 0)$. There exists a k -aligned triangulation $(G_T = (V_T, E_T), \mathcal{A})$ of (G, \mathcal{A}) whose size is $O(k^4 n)$. Each edge in $E_T \setminus E(G)$ is at most 1-crossed and 0-anchored, or 0-crossed and 1-anchored.*

Proof sketch. To triangulate (G, \mathcal{A}) , we exhaustively apply each of the following steps.

1. If f is a non-triangular face whose boundary contains a 2-crossed edge uv , we build a triangle by inserting a vertex x in the intermediate cell, as shown in Fig. 4(a). This step ensures that every edge of a non-triangular face is at most 1-crossed.
2. If f is a non-triangular face whose interior contains the intersection I of a set of pseudolines, we place a vertex v on I and connect v to two disjoint vertices on f with two simple paths, where every vertex of the path, which is not an endpoint, is free; compare Fig. 4(b).
3. If f is a non-triangular face with an aligned edge $e = uv$ we can split f into two faces f' and f'' (as shown in Fig. 4(c)) such that f' contains e on its boundary. Then we can triangulate f' with 1-crossed edges. A similar approach works for aligned vertices; see Fig. 4(d).
4. If f is a non-triangular face whose interior contains a pseudosegment \mathcal{S} , then we find two edges vw, xy as shown in Fig. 4(e) and we can triangulate by inserting a vertex on \mathcal{S} and 1-crossed edges.
5. If none of the cases above applies, then no non-triangular face contains a pseudosegment. Thus all remaining non-triangular faces can be triangulated with free edges. \square

In order to simplify the constructions we augment the input graph with an additional cycle in the outer face, so that no two pseudolines intersect in the outer face. More formally, let \mathcal{A} be an arrangement of pseudolines $\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_k$. Let $U_1, U_2, \dots, U_t \in \text{cells}(\mathcal{A})$ be the set of unbounded cells in the arrangement of \mathcal{A} such that U_i, U_{i+1} are adjacent cells with $U_{t+1} = U_1$. For $k > 1$, a k -aligned graph is *proper* (i) if the boundary of the outer face is a 0-anchored 1-crossed cycle of length t such that every unbounded region U_i contains exactly one vertex of the cycle, and (ii) every aligned edge in (G, \mathcal{A}) is 0-crossed. Observe that for every k -aligned graph (G, \mathcal{A}) there is proper k -aligned triangulation (G', \mathcal{A}) containing G as a minor.

The following two lemmas show that we can reduce the size of the aligned graph and obtain a drawing by merging two drawings or by unpacking an edge.

Lemma 4 (\star). *Let (G, \mathcal{A}) be a k -aligned triangulation. Let T be a separating triangle splitting G into subgraphs $G_{\text{in}}, G_{\text{out}}$ so that $G_{\text{in}} \cap G_{\text{out}} = T$ and G_{out} contains the outer face of G . Then, (i) $(G_{\text{out}}, \mathcal{A})$ and $(G_{\text{in}}, \mathcal{A})$ are k -aligned triangulations, and (ii) (G, \mathcal{A}) has an aligned drawing if and only if there exists a common line arrangement A such that $(G_{\text{out}}, \mathcal{A})$ has an aligned drawing (Γ_{out}, A) and $(G_{\text{in}}, \mathcal{A})$ has an aligned drawing (Γ_{in}, A) with the outer face drawn as $\Gamma_{\text{out}}[T]$.*

Lemma 5 (\star). *Let (G, \mathcal{A}) be a k -aligned triangulation of alignment complexity $(1, 0, \perp)$ and let e be a 0-anchored aligned edge or a free edge of G that is not an edge of a separating triangle. Then $(G/e, \mathcal{A})$ is a k -aligned triangulation of alignment complexity $(1, 0, \perp)$. Further, (G, \mathcal{A}) has an aligned drawing if $(G/e, \mathcal{A})$ has an aligned drawing.*

Proof sketch. Since u and v either both lie in the same cell or both in the interior of a pseudosegment, $(G/e, \mathcal{A})$ is an aligned triangulation.

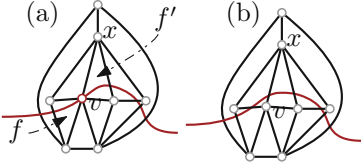


Fig. 5. Consistent transformation from a red vertex (a) to a gray vertex (b).

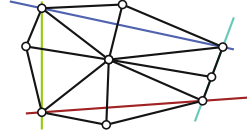


Fig. 6. All possible variations of vertices and edges in Lemma 7.

Let c be the vertex in G/e obtained by contracting $e = uv$ and let f be the face obtained by removing the vertex c from the aligned drawing (Γ', \mathcal{A}) of $(G/e, \mathcal{A})$. We place u at the position of c . This leaves a unique face f' to place v in. Since G/e is a triangulation, f' is star-shaped. Thus we can either place v close to u within its cell or on its line. \square

4.2 One Pseudoline

We show that every 1-aligned graph (G, \mathcal{R}) has an aligned drawing (Γ, R) , where \mathcal{R} is a single pseudoline and R the corresponding straight-line.

Lemma 6. *Let (G, \mathcal{R}) be a 1-aligned triangulation with k vertices on the outer face without a chord. If G is neither a triangle nor a k -wheel, then (G, \mathcal{R}) contains an interior aligned or an interior free edge.*

Proof. We first prove two useful claims.

Claim 1. Consider the order in which \mathcal{R} intersects the vertices and edges of G . If vertices u and v are consecutive on \mathcal{R} , then the edge uv is aligned.

Observe that the edge uv can be inserted into G without creating crossings. Since G is a triangulation, it therefore contains uv , and, further, since no 1-crossed edge can have both its endpoints on \mathcal{R} , it follows that indeed uv is aligned. This proves the claim.

Claim 2. Let (G, \mathcal{R}) be an aligned triangulation without aligned edges and let x be an interior free vertex of G , then x is incident to a free edge.

Assume for a contradiction that all neighbors of x lie either on \mathcal{R} or on the other side of \mathcal{R} . First, we slightly modify \mathcal{R} to a curve \mathcal{R}' that does not contain any vertices. Assume v is an aligned vertex; see Fig. 5. Since there are no aligned edges, \mathcal{R} enters v from a face f incident to v and leaves it to a different face f' incident to v . We then reroute \mathcal{R} from f to f' locally around v . If v is incident to x , we choose the rerouting such that it crosses the edge vx .

Notice that if e is intersected by \mathcal{R} in its endpoints, then \mathcal{R}' either does not intersect it, or intersects it in an interior point. Moreover, e cannot be intersected by \mathcal{R}' twice as in such case \mathcal{R} would pass through both its endpoints. Therefore (G, \mathcal{R}') is an aligned graph without any aligned vertices. Now, since G is a

triangulation, \mathcal{R}' is a simple cycle in the dual G^* of G , and hence corresponds to a cut C of G . Let H denote the connected component of $G - C$ that contains x and note that all edges of H are free. By the assumption and the construction of \mathcal{R}' , x is the only vertex in H . Thus, \mathcal{R}' intersects only the faces incident to x which are interior. This contradicts the assumption that \mathcal{R}' passes through the outer face of G . This finishes the proof of the claim.

We now prove the lemma. Assume that G is neither a triangle nor a k -wheel. Thus, G contains at least two interior vertices. If one of both vertices is free, we find a free edge by Claim 2. Otherwise, there is no free internal vertex, therefore the only edge which can intersect \mathcal{R} is a chord of G . Since G does not contain any chord, there is a pair of aligned vertices consecutive along \mathcal{R} . Thus by Claim 1 the instance (G, \mathcal{R}) has an aligned edge. \square

Theorem 3 (\star). *Let (G, \mathcal{R}) be an aligned graph and let (Γ_O, R) be a convex aligned drawing of the aligned outer face (O, \mathcal{R}) of G . There exists an aligned drawing (Γ, R) of (G, \mathcal{R}) with the same line R and the outer face drawn as Γ_O .*

Proof sketch. Given an arbitrary aligned graph (G, \mathcal{R}) , we first triangulate it using Lemma 3. As long as it has a free or an aligned edge e we do the following. If e is contained in a separating triangle, we decompose the graph using Lemma 4. Otherwise we simply contract e (Lemma 5). If no such edge exists, (G, \mathcal{R}) is either a triangle or a k -wheel (Lemma 6) and has an obvious straight-line aligned drawing. We obtain an aligned drawing of (G, R) by reversing the sequence contraction (Lemma 5) and decompositions along the separating triangles (Lemma 4). \square

4.3 Alignment Complexity $(1, 0, \perp)$

Let (G, \mathcal{A}) be a k -aligned graph of alignment complexity $(1, 0, \perp)$, i.e., every edge has at most one interior intersection and 2-anchored edges are forbidden. In this section, we prove that every such k -aligned graph has an aligned drawing. Figure 6 illustrates the statement of the following lemma.

Lemma 7. *Let (G, \mathcal{A}) be a proper k -aligned triangulation of alignment complexity $(1, 0, \perp)$ that does neither contain an interior free edge, nor a 0-anchored aligned edge, nor a separating triangle. Then (i) every intersection contains a vertex, (ii) every cell of the pseudoline arrangement contains exactly one free vertex, (iii) every pseudosegment is either covered by two aligned edges or intersected by an edge.*

Proof. The statement follows trivially from the following sequence of claims. We refer to an aligned vertex that is not an intersection vertex as a *flexible aligned* vertex.

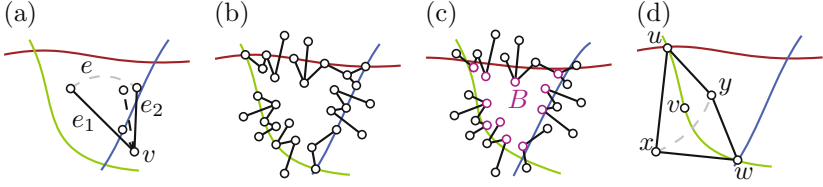


Fig. 7. Illustrations for the proof Lemma 7

Claim 1. Every pseudosegment alternately intersects flexible aligned vertices and edges.

Let \mathcal{S} be a pseudosegment in the pseudoline arrangement A . As in the proof of Lemma 6 one can argue that if two vertices occur consecutively along \mathcal{S} , then we find an aligned edge. Assume that \mathcal{S} intersects two edges e_1, e_2 consecutively as depicted in Fig. 7(a). Since G is a triangulation, it follows that e_1 and e_2 share an endpoint v . Every 1-crossed edge in G is 0-anchored, thus all endpoints of e_1 and e_2 must be free. Further e_1 and e_2 are consecutive in the circular order of edges around v as otherwise we would either find an intersection with \mathcal{S} between e_1 and e_2 or a free edge. Thus, e_1 and e_2 bound a face and are 1-crossed, hence their endpoints distinct from v are in the same cell and connected by an edge e , which is thus free. In a proper graph, the edges on the outer face are 1-crossed, thus, e is an interior edge, contradicting our assumptions.

Claim 2. Every cell contains at least one free vertex.

Observe that every triangle T containing the intersection of two pseudolines has at least one l -crossed edge, with $l \geq 2$. Since by definition (G, \mathcal{A}) does not contain 2-anchored aligned edges, T cannot contain an empty cell in its interior. Further, since G is proper, the outer face of G contains the intersection of every pair of pseudolines in its interior. Thus, since G is triangulated every cell contains at least one vertex.

Claim 3. Every cell contains at most one free vertex.

The following proof is similar to Claim 2 in the proof of Lemma 6. Let \mathcal{C} be a cell and assume for the sake of a contradiction that \mathcal{C} contains more than one vertex in the interior; see Fig. 7(b). These vertices are connected by edges to adjacent cells. If \mathcal{C} contains a vertex v on its boundary, we reroute the corresponding pseudolines close to v such that v is now outside of \mathcal{C} ; refer to Fig. 7(c). Let \mathcal{C}' be the resulting cell, it represents a cut in the graph with two components A and B , where \mathcal{C}' contains B in its interior. It is not difficult to see that the modified pseudolines are still pseudolines with respect to G . Since (G, \mathcal{A}) neither contains l -anchored edges nor l -crossed edges, $l \geq 2$, every edge of (G, \mathcal{A}) intersects the boundary of \mathcal{C}' at most once. Hence, B is connected and since it contains at least two vertices it also contains at least one free edge, contradicting our initial assumption.

Claim 4. Every flexible aligned vertex is incident to two 1-anchored aligned edges.

Let v be a vertex lying in the interior of a pseudosegment \mathcal{S} . Let u and w be the anchored vertices incident to \mathcal{S} . Further, let x and y be the vertices in the interior of the two cells incident to \mathcal{S} . Our instance (G, \mathcal{A}) is triangulated and every edge is at most 1-crossed. Thus, the vertices u, x, w, y build a quadrangle containing v in its interior. Since G does not contain a separating triangles, it cannot contain the edge xy . Moreover, \mathcal{S} contains exactly v in its interior, otherwise we would find a free aligned edge. Finally, since (G, \mathcal{A}) is an aligned triangulation, the vertex v is connected to all four vertices and, thus incident to two 1-anchored aligned edges.

Since (G, \mathcal{A}) is an aligned triangulation, Property (iii) immediately follows from Claims 3 and 4. \square

Lemma 8. *Let (G, \mathcal{A}) be a proper k -aligned triangulation of alignment complexity $(1, 0, \perp)$ that does neither contain an interior free edge, nor a 0-anchored aligned edge, nor a separating triangle. Let A be a line arrangement homeomorphic to the pseudoline arrangement \mathcal{A} . Then (G, \mathcal{A}) has an aligned drawing (Γ, A) .*

Proof. We obtain a drawing (Γ, A) by placing a free vertex in its cell, an aligned vertex on its pseudosegment and an intersection vertex on its intersection. According to Lemma 7 every cell and every intersection contains exactly one vertex and each pseudosegment is either crossed by an edge or it is covered by two aligned edges. Observe that the union of two adjacent cells of the arrangement A is convex. Thus, this drawing of G has an homeomorphic embedding to (G, \mathcal{A}) and every edge intersects in (Γ, A) the line $L \in A$ corresponding to the pseudoline $\mathcal{L} \in \mathcal{A}$ in (G, \mathcal{A}) \square

The following theorem can be proven along the same lines as Theorem 3.

Theorem 4 (\star). *Every k -aligned graph (G, \mathcal{A}) of alignment complexity $(1, 0, \perp)$ with a stretchable pseudoline arrangement \mathcal{A} has an aligned drawing.*

5 Conclusion

In this paper we showed that if \mathcal{A} is stretchable, then every k -aligned graph (G, \mathcal{A}) of alignment complexity $(1, 0, \perp)$ has a straight-line aligned drawing. As an intermediate result we showed that a 1-aligned graph (G, \mathcal{R}) has an aligned drawing with a fixed convex drawing of the outer face. We showed that the less restricted version of this problem, where we are only given a set of vertices to be aligned, is \mathcal{NP} -hard but fixed-parameter tractable.

The case of more general alignment complexities is widely open. Our techniques imply the existence of one-bend aligned drawings of general 2-aligned graphs [13]. However, the existence of straight-line aligned drawings is unknown even if in addition to 1-crossed edges, we only allow 2-anchored edges, i.e., in the case of alignment complexity $(1, 0, 0)$. In particular, there exist 2-aligned graphs

that neither contain a free edge nor an aligned edge but their size is unbounded in the size of the arrangement; see full version [12]. It seems that further reductions are necessary to arrive at a base case that can easily be drawn. This motivates the following questions.

- (1) What are all the combinations of line numbers k and alignment complexities C such that for every k -aligned graph (G, \mathcal{A}) of alignment complexity C there exists a straight-line aligned drawing provided \mathcal{A} is stretchable?
- (2) Given a k -aligned graph (G, \mathcal{A}) and a line arrangement A homeomorphic to \mathcal{A} , what is the complexity of deciding whether (G, \mathcal{A}) admits a straight-line aligned drawing (Γ, A) ?

References

1. Biedl, T., Kaufmann, M., Mutzel, P.: Drawing planar partitions II: HH-drawings. In: Hromkovič, J., Sýkora, O. (eds.) WG 1998. LNCS, vol. 1517, pp. 124–136. Springer, Heidelberg (1998). https://doi.org/10.1007/10692760_11
2. Cano, J., Tóth, C.D., Urrutia, J.: Upper bound constructions for untangling planar geometric graphs. *SIAM J. Discrete Math.* **28**(4), 1935–1943 (2014)
3. Chaplick, S., Fleszar, K., Lipp, F., Ravsky, A., Verbitsky, O., Wolff, A.: Drawing graphs on few lines and few planes. In: Hu, Y., Nöllenburg, M. (eds.) GD 2016. LNCS, vol. 9801, pp. 166–180. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_14
4. Chaplick, S., Fleszar, K., Lipp, F., Ravsky, A., Verbitsky, O., Wolff, A.: The complexity of drawing graphs on few lines and few planes. *Algorithms and Data Structures*. LNCS, vol. 10389, pp. 265–276. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62127-2_23
5. Da Lozzo, G., Dujmović, V., Frati, F., Mchedlidze, T., Roselli, V.: Drawing planar graphs with many collinear vertices. In: Hu, Y., Nöllenburg, M. (eds.) GD 2016. LNCS, vol. 9801, pp. 152–165. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_13
6. Dujmović, V.: The utility of untangling. In: Di Giacomo, E., Lubiw, A. (eds.) GD 2015. LNCS, vol. 9411, pp. 321–332. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27261-0_27
7. Dujmović, V., Evans, W., Kobourov, S., Liotta, G., Weibel, C., Wismath, S.: On graphs supported by line sets. In: Brandes, U., Cornelsen, S. (eds.) GD 2010. LNCS, vol. 6502, pp. 177–182. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-18469-7_16
8. Dujmović, V., Langerman, S.: A center transversal theorem for hyperplanes and applications to graph drawing. *Discrete Comput. Geom.* **49**(1), 74–88 (2013)
9. Fößmeier, U., Kaufmann, M.: Nice drawings for planar bipartite graphs. In: Bongiovanni, G., Bovet, D.P., Di Battista, G. (eds.) CIAC 1997. LNCS, vol. 1203, pp. 122–134. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-62592-5_66
10. Goodman, J.E., Pollack, R.: Proof of Grünbaum’s conjecture on the stretchability of certain arrangements of pseudolines. *J. Comb. Theory Ser. A* **29**(3), 385–390 (1980)
11. Levi, F.: Die Teilung der projektiven Ebene durch Gerade oder Pseudogerade. *Ber. Math.-Phys. Kl. Sächs. Akad. Wiss* **78**, 256–267 (1926)

12. Mchedlidze, T., Radermacher, M., Rutter, I.: Aligned Drawings of Planar Graphs (2017). <https://arxiv.org/abs/1708.08778v2>
13. Mchedlidze, T., Radermacher, M., Rutter, I.: Aligned drawings of planar graphs. In: Proceedings of the 33rd European Workshop on Computational Geometry (EuroCG 2017) (2017)
14. Mnev, N.E.: The universality theorems on the classification problem of configuration varieties and convex polytopes varieties. In: Viro, O.Y., Vershik, A.M. (eds.) *Topology and Geometry — Rohlin Seminar*. LNM, vol. 1346, pp. 527–543. Springer, Heidelberg (1988). <https://doi.org/10.1007/BFb0082792>
15. Shor, P.: Stretchability of pseudolines is NP-hard. In: *Applied Geometry and Discrete Mathematics-The Victor Klee Festschrift*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 4, pp. 531–554. American Mathematical Society (1991)
16. Wahlström, M.: Abusing the Tutte matrix: an algebraic instance compression for the K-set-cycle problem. In: *30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013*, pp. 341–352 (2013)

On the Edge-Length Ratio of Outerplanar Graphs

Sylvain Lazard¹, William Lenhart², and Giuseppe Liotta³(✉)

¹ Inria, CNRS, University of Lorraine, Nancy, France
sylvain.lazard@inria.fr

² Williams College, Williamstown, USA
wlenhart@williams.edu

³ University of Perugia, Perugia, Italy
giuseppe.liotta@unipg.it

Abstract. We show that any outerplanar graph admits a planar straight-line drawing such that the length ratio of the longest to the shortest edges is strictly less than 2. This result is tight in the sense that for any $\epsilon > 0$ there are outerplanar graphs that cannot be drawn with an edge-length ratio smaller than $2 - \epsilon$. We also show that every bipartite outerplanar graph has a planar straight-line drawing with edge-length ratio 1, and that, for any $k \geq 1$, there exists an outerplanar graph with a given combinatorial embedding such that any planar straight-line drawing has edge-length ratio greater than k .

1 Introduction

The problem of computing a planar straight-line drawing with prescribed edge lengths has been addressed by several authors, partly for its theoretical interest and partly for its application in different areas, including VLSI, wireless sensor networks, and computational geometry (see for example [6, 7, 9, 13]). Deciding whether a graph admits a straight-line planar drawing with prescribed edge lengths was shown to be NP-hard by Eades and Wormald for 3-connected planar graphs [8]. In the same paper, the authors show that it is NP-hard to determine whether a 2-connected planar graph has a *unit-length* planar straight-line drawing; that is, a drawing in which all edges have the same length. Cabello et al. extend this last result by showing that it is NP-hard to decide whether a 3-connected planar graph admits a unit-length planar straight-line drawing [3]. In addition, Bhatt and Cosmadakis prove that deciding whether a degree-4 tree has a planar drawing such that all edges have the same length and the vertices are at integer grid points is also NP-hard [2].

These hardness results have motivated the study of relaxations and variants of the problem of computing straight-line planar drawings with constraints on the edge lengths. For example, Aichholzer et al. [1] study the problem of computing straight-line planar drawings where, for each pair of edges of the input graph G , it is specified which edge must be longer. They characterize families of graphs

that are *length universal*, i.e. they admit a planar straight-line drawing for any given total order of their edge lengths.

Perhaps one of the most natural variants of the problem in the context of graph drawing is that where, instead of imposing constraints on the edge lengths, one aims at computing planar straight-line drawings where the variance of the lengths of the edges is minimized. See for example [5], where this optimization goal is listed among the most relevant aesthetics that impact the readability of a drawing of a graph. Computing straight-line drawings where the ratio of the longest to the shortest edge is close to 1 also arises in the approximation of unit disk graph representations, a problem of interest in the area of wireless communication networks (see, e.g. [4, 11]).

Discouragingly, Eades and Wormald observe in their seminal paper that the NP-hardness of computing 2-connected planar straight-line drawings with unit edge lengths persists even when a small tolerance (independent of the problem size) in the length of the edges is allowed. To our knowledge, little progress has been made on bounding the ratio between the longest and shortest edge lengths in planar straight-line drawings. We recall the work of Hoffmann et al. [10], who compare different drawing styles according to different quality measures including the edge-length variance.

In this paper we study planar straight-line drawings of outerplanar graphs that bound the ratio of the longest to the shortest edge lengths from above by a constant. We define the *planar edge-length ratio* of a planar graph G as the smallest ratio between the longest and the shortest edge lengths over all planar straight-line drawings of G . The main result of the paper is the following.

Theorem 1. *The planar edge-length ratio of an outerplanar graph is strictly less than 2. Also, for any given real positive number ϵ , there exists an outerplanar graph whose planar edge-length ratio is greater than $2 - \epsilon$.*

Informally, Theorem 1 establishes that 2 is a tight bound for the planar edge-length ratio of outerplanar graphs. The upper bound is proved by using a suitable decomposition of an outerplanar graph into subgraphs called *strips*, then drawing the graph strip by strip. The lower bound is proved by taking into account all possible planar embeddings of a maximal outerplanar graph whose maximum vertex degree is a function of ϵ . Theorem 1 naturally suggests some interesting questions that are discussed in Sect. 3.

We shall assume familiarity with basic definitions of graph planarity and of graph drawing [5] and introduce only the terminology and notation that is strictly needed for our proofs.

2 Proof of Theorem 1

It suffices to establish the result for maximal outerplanar graphs. To show that the edge-length ratio of a maximal outerplanar graph is always less than 2, we imagine decomposing the dual G^* of G into a set of disjoint paths, which we call *chains*. Each chain corresponds to some sequence of pairwise-adjacent

triangles of G . The set of chains inherits a tree structure from G^* , and we use this structure to direct an algorithm that draws each of the chains proceeding from the root of this tree down to its leaves. We formally define a chain to be a sequence T_s, T_{s+1}, \dots, T_t of triangles of G where $s \leq 0 \leq t$, and such that (i) T_0 consists of an outer edge of G whose vertices are labeled with 0, along with a third vertex labeled 1, (ii) for each $i : 1 \leq i \leq t$, the vertices of T_i are labeled by $\{i - 1, i, i + 1\}$ so that T_i and T_{i-1} share the edge having vertices labeled i and $i - 1$, and (iii) for each $i : -1 \geq i \geq s$, the vertices of T_i are labeled by $\{i, i + 1, i + 2\}$ so that T_i and T_{i+1} share the edge having vertices labeled $i + 1$ and $i + 2$. Note that this definition prohibits *fans* (consecutive triangles all sharing a common vertex) containing more than 3 triangles, except for the vertex labeled 1, which has four incident triangles on the chain.

The decomposition into chains is constructed by first selecting an edge e' on the outer face of some outerplanar topological embedding of G . The edge e' is incident with a unique triangle of G . Label each vertex of e' with 0, and label the third vertex of the triangle with 1. There is now a unique maximal chain $C_{e'}$ in G containing this labeled triangle. The edges of $C_{e'}$ can be partitioned into two sets: $S_{e'}$ and $L_{e'}$ where $S_{e'}$ consists of edges of $C_{e'}$ whose vertex labels differ by 1 and $L_{e'}$ consists of all edges of $C_{e'}$ whose vertex labels differ by 2, along with e' .

Removing the edges of $S_{e'}$ from G produces a set of 2-connected components in 1-1 correspondence with the edges of $L_{e'}$: Each component contains exactly one element of $L_{e'}$ which lies on its outer face. For each edge $e \in L_{e'}$, let G_e be the component of $G - S_{e'}$ containing e . We can then recursively decompose each G_e by choosing the (unique) maximal chain C_e in G_e containing the one triangle (if any) of G_e that is incident with e . We call the set of chains so constructed a *chain decomposition of G* . A chain decomposition produces a decomposition of the edges of G into sets L and S , where L is the union of the edges in each L_e and S is the union of all of the edges in each S_e . Note that there is a single chain for each edge in L , and that the collection of chains produced naturally form a tree: The root of the tree is the chain $C_{e'}$ and its children are the chains $\{C_e : e \in L_{e'}\}$; the chain decomposition is entirely determined by the choice of external edge e' .

The drawing algorithm proceeds by first drawing the root chain $C_{e'}$ of the chain decomposition tree of G and then recursively drawing the chain decomposition trees of each $G_e : e \in L_{e'}$. The algorithm depends on a specific method for drawing a single chain. To describe it, we need a few definitions. First, given a line segment s in the plane and a direction (unit vector) \mathbf{d} not parallel to s , denote by $S(s, \mathbf{d})$ the half-infinite strip bounded by s and the two infinite rays in direction \mathbf{d} that have their sources at the endpoints of s . Finally, given a chain C , the edges of $C \cap L$ are called *external edges* of C ; note that each external edge is incident to exactly one triangle of C .

Lemma 1. *Given a chain C with n vertices, an external edge e of C , a segment s of length 1 in the plane, and a direction \mathbf{d} such that the (smaller) angle between s and \mathbf{d} is $\theta < \theta_0 = \arccos(1/4) \approx 75.5^\circ$, there exists a planar straight-line drawing*

of C such that: (i) the drawing is completely contained within the strip $S(s, \mathbf{d})$; (ii) no external edge e' of C is parallel to \mathbf{d} , and the strips $S(e', \mathbf{d})$ are all empty; (iii) each external edge has length 1 and all other edges have lengths greater than $1/2$; (iv) each external edge forms an angle less than θ_0 with \mathbf{d} . Moreover, such a drawing can be computed in $O(n)$ -time in the real RAM model.

Proof. Let T_0 be the triangle of C containing e . T_0 is either adjacent to zero, one, or two triangles of C . We handle these three cases in turn. If T_0 is the only triangle in C , then we simply draw T_0 in $S(s, \mathbf{d})$ as an isosceles triangle with e drawn as s and with its third vertex drawn so that its two edges have length l , where $\frac{1}{2} < l < 1$.

Assume now that T_0 is adjacent to a triangle T_1 of C . Denote the vertices of C as follows: $T_0 = \{v_0^-, v_0^+ v_1\}$, where $e = \{v_0^-, v_0^+\}$ and v_0^- is not incident with T_1 . The vertex of T_1 not in T_0 is denoted by v_2 , and, subsequently, the vertex of each T_i not in T_{i-1} is denoted by v_{i+1} . We draw T_0 as previously, but with more careful positioning of v_1 . To determine where to position v_1 , we draw edge $\{v_0^+, v_2\}$ of T_1 as a unit-length segment in direction \mathbf{d} . As long as v_1 is positioned within $S(s, \mathbf{d})$ but outside of the disks of radius $\frac{1}{2}$ centered at v_0^-, v_0^+ , and v_2 , the edges from each of these vertices to v_1 will have length greater than $\frac{1}{2}$ (see top half of Fig. 1). By placing v_1 close to e , the edges $\{v_1, v_0^-\}$ and $\{v_1, v_0^+\}$ will have lengths less than 1. Also, since $\angle v_2 v_0^+ v_1 < \theta_0$, edge $\{v_1, v_2\}$ will have length less than 1.

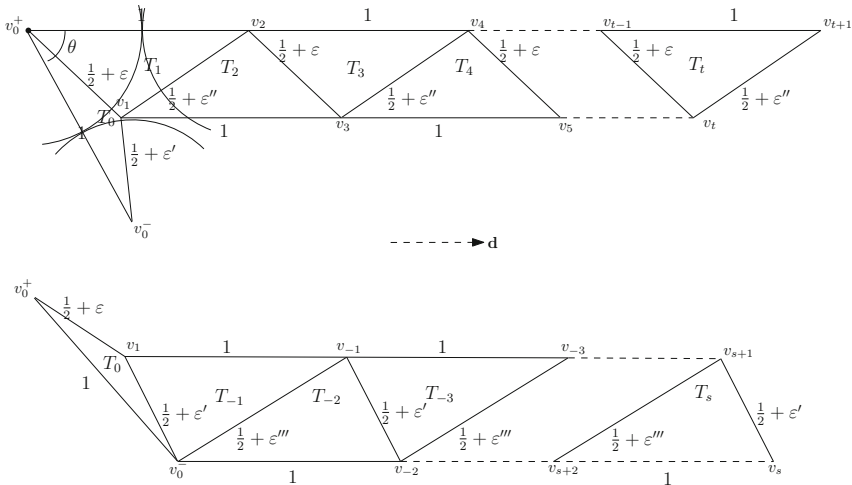


Fig. 1. Drawing a chain

Assuming that T_0, \dots, T_{i-1} have been drawn for some $i > 1$, T_i is drawn by positioning v_{i+1} one unit distant from v_{i-1} in direction \mathbf{d} . The result is that each T_i is congruent to T_1 and so the edge-length ratio of C is less than 2.

At this point, all of the unit-length segments, except for e , lie on the two rays in direction \mathbf{d} emanating from v_0^+ and v_1 . By rotating these rays a very small amount towards one another, we can preserve the lengths of the unit-length segments while ensuring that all of the remaining segments have lengths in the range $(\frac{1}{2}, 1)$. See the top of Fig. 1.

Finally, suppose that T_0 has two adjacent triangles. Starting with T_0 , label the other triangles in C so that the labels of adjacent triangles differ by 1. Thus, for example, T_0 is adjacent to T_1 and T_{-1} . The vertices in the $T_i, i \geq 0$ will be labeled as in the previous case: the unique vertex in T_i not in T_{i-1} is labeled v_{i+1} . The vertices in the $T_i, i < 0$ will be similarly labeled: the unique vertex in T_i not in T_{i+1} is labeled v_i .

Draw each $T_i, i > 0$ as in the previous case. Now draw the $T_i, i < 0$ in a similar fashion: Place vertex v_i one unit distant from v_{i+2} in direction \mathbf{d} . Then, as above, all of the unit length edges of the triangles $T_i, i < 0$ will lie on the two rays in direction \mathbf{d} emanating from v_1 and v_0^- , and these two rays can be rotated slightly towards each other while maintaining the length of the unit-length edges and ensuring that the other edges still have lengths in the range $(\frac{1}{2}, 1)$. See the bottom of Fig. 1. This can clearly be done so that all external edges form angles less than θ_0 with \mathbf{d} .

However, we need to ensure that v_1 can be placed so that both triangle T_1 and triangle T_{-1} can simultaneously satisfy the required edge-length conditions: Namely, that edges $\{v_0^-, v_0^+\}$, $\{v_1, v_{-1}\}$, and $\{v_0^+, v_2\}$ are all unit-length, while edges $\{v_1, v_2\}$, $\{v_0^+, v_1\}$, $\{v_1, v_0^-\}$, and $\{v_0^-, v_{-1}\}$ all have lengths in the range $(\frac{1}{2}, 1)$. However, it is relatively simple to show that v_1 can always be successfully placed if the (smaller) angle between s and \mathbf{d} is less than $\theta_0 = \arccos(1/4)$. [The angle θ_0 is the angle opposite an edge of length 2 in an isosceles triangle having side lengths 2, 2, and 1.] Finally, the computation of the locations of the vertices can each be computed in constant time in the real RAM model, giving a run-time linear in the size of the chain. \square

We are now ready to prove the following lemma. For a planar straight-line drawing Γ , we denote with $\rho(\Gamma)$ the ratio of a longest to a shortest edge in Γ .

Lemma 2. *A maximal outerplanar graph G with n vertices admits a planar straight-line drawing Γ with $\rho(\Gamma) < 2$ that can be computed in $O(n)$ time assuming the real RAM model of computation.*

Proof. We call the drawing computed as in Lemma 1 a *U-strip drawing* of C and adopt the same notation as in Lemma 1. Recall that in a chain decomposition of a graph, the external edges of the chains are exactly the edges of L . A drawing of G is computed as follows.

1. Compute a chain decomposition tree for G ; let $C_{e'}$ be the root of the tree.
2. Select a line segment s of length 1 in the plane and an initial direction \mathbf{d} not parallel to s such that $\angle s\mathbf{d} < \theta_0$.
3. Apply Lemma 1 to compute a U-strip drawing of $C_{e'}$.

4. Each edge $e \in L_{e'}$ is drawn as a segment s_e of length 1, not parallel to \mathbf{d} , that forms an angle with \mathbf{d} that is less than θ_0 , so draw the subtree of $C_{e'}$ rooted at C_e in the empty strip $S(s_e, \mathbf{d})$.

The result is an outerplanar straight-line drawing in which all edges of L (*long edges*) have length 1 while all edges in S (*short edges*) have length strictly greater than $\frac{1}{2}$. If we assume that the input is provided to the algorithm in the form of a doubly-connected edge list [12], then a chain decomposition tree for G can be computed in linear time. Also, since by Lemma 1 each chain can be drawn in time proportional to its length, the algorithm runs in $O(n)$ time in the real RAM model. \square

The following lemma can be proved by means of a packing argument and elementary geometry.

Lemma 3. *For any $\epsilon > 0$ there exists a maximal outerplanar graph whose planar edge length ratio is greater than $2 - \epsilon$.*

We conclude the section by observing that Lemmas 2 and 3 imply Theorem 1.

3 Additional Remarks and Open Problems

The upper and the lower bound of Theorem 1 suggest some questions that we find worth investigating. One question is whether better bounds on the planar edge-length ratio can be established for subfamilies of outerplanar graphs (for example, it is easy to show that trees have unit-length drawings). A second question is whether an edge length variance bounded by a constant can be guaranteed for drawings of outerplanar graphs where not all vertices lie in a common face. By a variant of the approach used to prove Lemma 2 and by using some simple geometric observations, the following results can be proved.

Theorem 2. *The planar edge-length ratio of a bipartite outerplanar graph is 1.*

The *plane edge-length ratio* of a planar embedding \mathcal{G} of a graph G is the minimum edge-length ratio taken over all embedding-preserving planar straight-line drawings of \mathcal{G} .

Theorem 3. *For any given $k \geq 1$, there exists an embedded outerplanar graph whose plane edge-length ratio is at least k .*

We conclude this paper by listing some open questions that we find interesting to study: (i) Study the edge-length ratio of triangle-free outerplanar graphs. For example, it is not hard to see that if all faces of an outerplanar graph have five vertices, a unit edge length drawing may not exist; however, the planar edge length ratio for this family of graphs could be smaller than the one established in Theorem 1. (ii) Extend the result of Theorem 1 to families of non-outerplanar graphs. For example it would be interesting to study whether the planar edge-length ratio of 2-trees is bounded by a constant. (iii) Study the complexity of deciding whether an outerplanar graph admits a straight-line drawing where the ratio of the longest to the shortest edge is within a given constant. This problem is interesting also in the special case that we want all edges to be unit length.

References

1. Aichholzer, O., Hoffmann, M., van Kreveld, M.J., Rote, G.: Graph drawings with relative edge length specifications. In: 2014 Proceedings of the 26th Canadian Conference on Computational Geometry, CCCG 2014, Halifax, Nova Scotia, Canada (2014)
2. Bhatt, S.N., Cosmadakis, S.S.: The complexity of minimizing wire lengths in VLSI layouts. *Inf. Process. Lett.* **25**(4), 263–267 (1987)
3. Cabello, S., Demaine, E.D., Rote, G.: Planar embeddings of graphs with specified edge lengths. *J. Graph Algorithms Appl.* **11**(1), 259–276 (2007)
4. Chen, J., Jiang, A.A., Kanj, I.A., Xia, G., Zhang, F.: Separability and topology control of quasi unit disk graphs. *Wirel. Netw.* **17**(1), 53–67 (2011)
5. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, Upper Saddle River (1999)
6. Di Battista, G., Vismara, L.: Angles of planar triangular graphs. *SIAM J. Discrete Math.* **9**(3), 349–359 (1996)
7. Doherty, L., Pister, K.S.J., El Ghaoui, L.: Convex optimization methods for sensor node position estimation. In: *Proceedings IEEE INFOCOM 2001, The Conference on Computer Communications, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Twenty years into the communications odyssey, Anchorage, Alaska, USA, 22–26 April 2001*, pp. 1655–1663. IEEE (2001)
8. Eades, P., Wormald, N.C.: Fixed edge-length graph drawing is NP-hard. *Discrete Appl. Math.* **28**(2), 111–134 (1990)
9. Held, S., Korte, B., Rautenbach, D., Vygen, J.: Combinatorial optimization in VLSI design. In: Chvátal, V. (ed.) *Combinatorial Optimization - Methods and Applications*, NATO Science for Peace and Security Series - D: Information and Communication Security, vol. 31, pp. 33–96. IOS Press (2011)
10. Hoffmann, M., van Kreveld, M.J., Kusters, V., Rote, G.: Quality ratios of measures for graph drawing styles. In: 2014 Proceedings of the 26th Canadian Conference on Computational Geometry, CCCG 2014, Halifax, Nova Scotia, Canada (2014)
11. Kuhn, F., Moscibroda, T., Wattenhofer, R.: Unit disk graph approximation. In: Basagni, S., Phillips, C.A. (eds.) *Proceedings of the DIALM-POMC Joint Workshop on Foundations of Mobile Computing*, Philadelphia, PA, USA, 1 October 2004, pp. 17–23. ACM (2004)
12. Muller, D.E., Preparata, F.P.: Finding the intersection of two convex polyhedra. *Theoret. Comput. Sci.* **7**(2), 217–236 (1978)
13. Savarese, C., Rabaey, J.M., Langendoen, K.: Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In: Ellis, C.S. (ed.) *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*, Monterey, California, USA, 10–15 June 2002, pp. 317–327. USENIX (2002)

On Vertex- and Empty-Ply Proximity Drawings

Patrizio Angelini¹(✉), Steven Chaplick², Felice De Luca³, Jiří Fiala⁴,
Jaroslav Hančl Jr.⁴, Niklas Heinsohn¹, Michael Kaufmann¹,
Stephen Kobourov⁵, Jan Kratochvíl⁴, and Pavel Valtr⁴

¹ Wilhelm-Schickhard-Institut für Informatik, Universität Tübingen,
Tübingen, Germany

angelini@informatik.uni-tuebingen.de

² Lehrstuhl für Informatik I, Universität Würzburg, Würzburg, Germany

³ Università degli Studi di Perugia, Perugia, Italy

⁴ Department of Applied Mathematics, Charles University (KAM),
Prague, Czech Republic

⁵ Department of Computer Science, University of Arizona, Tucson, USA

Abstract. We initiate the study of the *vertex-ply* of straight-line drawings, as a relaxation of the recently introduced *ply* number. Consider the disks centered at each vertex with radius equal to half the length of the longest edge incident to the vertex. The vertex-ply of a drawing is determined by the vertex covered by the maximum number of disks. The main motivation for considering this relaxation is to relate the concept of ply to proximity drawings. In fact, if we interpret the set of disks as proximity regions, a drawing with vertex-ply number 1 can be seen as a weak proximity drawing, which we call *empty-ply* drawing. We show non-trivial relationships between the ply number and the vertex-ply number. Then, we focus on empty-ply drawings, proving some properties and studying what classes of graphs admit such drawings. Finally, we prove a lower bound on the ply and the vertex-ply of planar drawings.

1 Introduction

Constructing graph layouts that are readable and easily convey the information hidden in the represented data is one of the main goals of graph drawing research. Several aesthetic criteria have been defined to capture the user requirement for a better understanding of the data, e.g., resolution rules [13, 18], low-density [14], proximity drawings [17]. The *ply number* [10] of a graph is another such criterion. We adopt the following notation: given a straight-line drawing Γ of a graph $G = (V, E)$, for each vertex $v \in V$ consider an open disk D_v (called the *ply-disk* of v) centered at v with radius r_v equal to half of the length of the longest edge incident to v . Over all points p on the plane, let k be the maximum number of ply-disks of Γ that include the point p in their interior. Then, the drawing Γ has *ply* k . The *ply number* of G is the minimum ply over all its drawings.

The ply number was originally proposed by Eppstein and Goodrich [12] in the context of interpreting road networks as subgraphs of disk-intersection graphs.

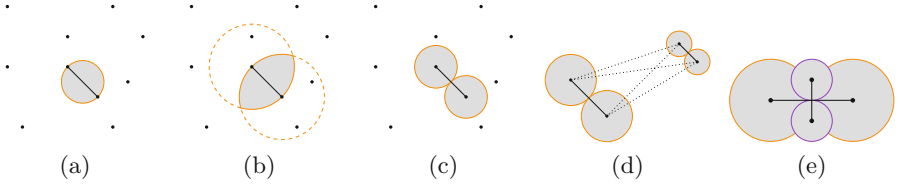


Fig. 1. (a) Gabriel, (b) Relative-neighborhood, and (c) Ply proximity regions. (d) A disconnected empty-ply graph. (e) A non-planar empty-ply drawing.

The concept of a ply number is also related to proximity drawings of graphs [17]. A *proximity drawing* of a graph G is a straight-line drawing of G in which for every two vertices u and v , there exists a region of the plane, called *proximity region* of u and v , that contains other vertices in its interior if and only if u and v are not connected by an edge in G . If G admits a proximity drawing, then it is a *proximity graph*. A proximity region specifies a set of points in the plane that are closer to u and v than to the other vertices, and different proximity regions lead to different definitions of proximity drawings. Regions can be *global*, e.g., Euclidean minimum spanning trees [19], or *local*, e.g., Gabriel graphs [15] (Fig. 1a), relative-neighborhood graphs [20] (Fig. 1b), and Delaunay triangulations [8, 19]. Proximity drawings of graphs are also studied in the *weak* model [9], where the “if” part of the condition is neglected: i.e., if two vertices are not connected by an edge, then their proximity region may be empty.

In this work, we are interested in deepening the study of the relationship between the notions of ply number and of proximity drawings. In this direction, one can consider the local proximity region associated with a pair of vertices u and v as the one composed of the disks centered at u and at v , with radius equal to half of the length of the straight-line segment between u and v (Fig. 1c). Due to the possible absence of edges, this is a weak proximity model. However, a drawing Γ may have ply larger than 1 even if no proximity region contains a vertex different from the two which defined it, since the ply of Γ is only determined by the way in which different regions intersect each other.

To improve this relationship, we relax the definition of ply number and introduce the concept of *vertex-ply number*. Consider a straight-line drawing Γ of a graph G . Over all vertex-points p on the plane (i.e., points which realize a vertex of G), let k be the maximum number of ply-disks of Γ that include the point p in their interior. Then, the drawing Γ has *vertex-ply* k . The *vertex-ply number* of G is the minimum vertex-ply over all its drawings. In the special case in which Γ has vertex-ply 1, i.e., every disk D_v contains only v in its interior, we say that Γ is an *empty-ply* drawing. Note that an empty-ply drawing is in fact a weak proximity drawing with respect to the proximity region defined above, that is, a drawing is empty-ply if and only if all the proximity regions are empty.

Some relationships between proximity models are known, e.g., any Delaunay triangulation contains a Gabriel graph as a spanning subgraph, which in turn contains a relative-neighborhood graph, which in turn contains a

minimum spanning tree [17]. It is hence natural to ask about the role of empty-ply drawings in these relationships. We first note that an empty-ply drawing may be non-planar (see Fig. 1e), which is not the case for Delaunay triangulations, and thus for any of the other type of proximity drawings. On the other hand, there exist empty-ply drawings that are not connected and that cannot be made connected by just adding edges while maintaining the empty-ply property (see Fig. 1d), which differs from the case for minimum spanning trees (and thus for all the other proximity drawings). These two observations imply that empty-ply drawings are not directly comparable with other types of disk-based proximity drawings.

The concept of empty-ply is related to *partial edge drawings* (PEDs) [4–6]. A PED is a straight-line drawing of a graph in which each edge is divided into three segments: a *middle part* that is not drawn and the two segments incident to the vertices, called *stubs*, that remain in the drawing and that are not allowed to cross. Our Theorem 2 in Sect. 3 shows that an empty-ply drawing also yields a PED whose stubs have nontrivial lengths.

Drawing graphs with low ply was first considered by Di Giacomo *et al.* [10]. They show that testing whether an internally triangulated biconnected planar graph has ply number 1 can be done in $O(n \log n)$ time and that the class of graphs with ply number 1 coincides with unit-disk contact graphs [3], which makes the recognition problem NP-hard. Angelini *et al.* [1] studied area requirements of drawings of trees with low ply. De Luca *et al.* [7] performed an experimental study demonstrating correlations between the ply of a drawing and aesthetic metrics such as stress and uniform edge-lengths. An interactive tool has been implemented by Heinsohn and Kaufmann [16].

We first demonstrate non-trivial relationships between the ply number and the vertex-ply number of graphs. In Sect. 2 we positively answer a question from [10] (Problem 4) regarding whether the ply number of an empty-ply drawing is constant. In Sect. 3 we study properties of empty-ply graphs. In Sect. 4 we provide several classes of graphs that admit empty-ply drawings and some classes that do not (we consider k -ary trees, complete (bipartite) graphs, and squares of graphs with ply number 1). Further, in Sect. 5 we answer another question posed in [10] (Problem 3), regarding the relationship between (vertex-) ply and crossings, by presenting graphs that admit drawings with constant ply and only 3 crossings but any corresponding planar drawing requires linear ply. We conclude in Sect. 6 with several open problems. For space reasons, some proofs have been sketched or omitted. Complete proofs can be found in the full version of the paper [2].

2 Relationships Between Ply and Vertex-Ply

We start with a natural question about the relationship between the ply number and the vertex-ply number of a graph.

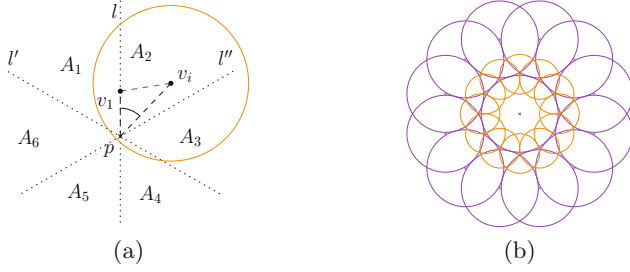


Fig. 2. (a) Illustration for the proof of Theorem 1. (b) An empty-ply drawing of a star of degree 24. For readability, edges are not drawn.

Theorem 1. *The ply of a drawing of a graph with vertex-ply h is at most $5h$.*

Proof. Let Γ be any drawing of a graph G with vertex-ply h . Let p be any point in the plane and let v_1, \dots, v_k be the vertices whose ply-disks contain p in their interior, appearing in this radial order around p ; see Fig. 2a. Without loss of generality, assume that v_1 is the vertex closest to p . Let l be the line through p and v_1 , and let l' and l'' be two lines through p creating angles $\frac{\pi}{3}$ and $-\frac{\pi}{3}$ with l . These lines determine a covering of the plane by six closed wedges A_1, \dots, A_6 centered at p , each having $\frac{\pi}{3}$ as its internal angle.

Let A_1 and A_2 be the wedges delimited by the half-line starting at p and passing through v_1 . For each vertex $v_i \in A_1 \cup A_2$ we have $\angle v_1 p v_i \leq \frac{\pi}{3}$. This implies that $|v_1 v_i| \leq |v_i p|$ and hence that v_1 belongs to the ply-disk D_{v_i} , since p belongs to D_{v_i} . Thus, if the union of the closed wedges A_1 and A_2 contains at least h vertices among v_2, \dots, v_k , we obtain that v_1 belongs to at least $h + 1$ ply-disks. This is not possible, since Γ has vertex-ply h .

We now prove that each wedge A_i with $3 \leq i \leq 6$ contains at most h vertices among v_2, \dots, v_k . Namely if it contains at least $h + 1$ vertices we can argue as above that the closest vertex to p among them belongs to the ply-disks of all the other h vertices. This completes the proof of the theorem that there exist at most $5h$ vertices whose ply-disks enclose p . \square

Corollary 1. *The ply of an empty-ply drawing of a graph is at most 5.*

Note that the converse of Corollary 1 does not hold. If a graph G does not admit any empty-ply drawing, that does not imply that the ply number of G is larger than 5. A star graph with degree larger than 24 does not have an empty-ply drawing (see Theorem 3), but can be drawn with constant ply 2 [10].

3 Properties of Graphs with Empty-Ply Drawings

Let Γ be a straight-line drawing of a graph G . Let $\{D'_v, v \in V\}$ be the set of open disks where D'_v is centered at v , but with radius only $\frac{r_v}{2}$. We can think of

these disks as obtained by shrinking the original ply-disks of Γ to half-length radius. Note that if Γ is an empty-ply drawing, then the disks in $\{D'_v, v \in V\}$ are pairwise disjoint. This observation implies the next result.

Lemma 1. *In an empty-ply drawing Γ of a graph $G = (V, E)$ the sum of the areas of all ply-disks $\{D'_v, v \in V\}$ does not exceed 4 times the area of their union.*

Proof. Each disk D'_v has area four times smaller than D_v , but is drawn inside the union of all ply-disks.

In the rest of the paper we frequently use disk-packing arguments based on Lemma 1. Another consequence of the observation above is a relationship between empty-ply drawings and the most popular type of PED, called $\frac{1}{4}$ -SHPED [5], in which the length of both stubs of an edge e is $\frac{1}{4}$ of e 's length.

Theorem 2. *An empty-ply graph admits a $\frac{1}{4}$ -SHPED.*

Proof. Let Γ be an empty-ply drawing of a graph $G = (V, E)$ with the set of disks $\{D'_v, v \in V\}$. Let Γ' be the drawing obtaining from Γ by keeping for each edge (u, v) only the two parts in the interior of disks D'_u and D'_v . By definition, both these parts cover at least $\frac{1}{4}$ of (u, v) . Since no two such disks overlap, there is no crossing in Γ' , and the statement follows. \square

We now focus on the relationship between the radii of the ply-disks of adjacent vertices in an empty-ply drawing. For the following two lemmas we use that for each vertex v , and for each edge (v, w) incident to v , we have $r_v \leq |vw|$, as the drawing is empty-ply, and $r_v \geq \frac{|vw|}{2}$, by the definition of the ply-disk D_v .

Lemma 2. *In an empty-ply drawing, for any two edges (u, v) and (v, w) incident to the same vertex v , we have $\frac{1}{2} \leq \frac{|uv|}{|vw|} \leq 2$.*

Lemma 3. *In an empty-ply drawing, the radii of the ply-disks of two adjacent vertices u and v differ by at most a factor of 2, i.e., $\frac{1}{2} \leq \frac{r_u}{r_v} \leq 2$.*

We conclude the section by presenting a tight bound on the maximum degree of graphs that admit empty-ply drawings.

Theorem 3. *No vertex of an empty-ply graph has degree greater than 24.*

Proof. To obtain a contradiction, let Γ be an empty-ply drawing of a graph G with a vertex v of degree greater than 24. By Lemma 2, the lengths of all edges of v are in the interval $[m, 2m]$, where m is the length of the shortest edge. Note that there are at least 13 edge lengths either in the interval $[m, \sqrt{2}m]$ or in the interval $[\sqrt{2}m, 2m]$. In either case, there exist two neighbors u and w of v such that $|vu| \leq |vw| \leq \sqrt{2}|vu|$ and $\alpha = \angle uvw \leq \frac{2\pi}{13}$. Scaling Γ by a factor of $|vu|^{-1}$, we may assume w.l.o.g. that $|vu| = 1$ and that $|vw| = q \in [1, \sqrt{2}]$. By the law of cosines, $|uw|^2 = 1 + q^2 - 2q \cos \alpha$. As Γ is an empty-ply drawing, the vertex v does not belong to the open disk α centered at w . Hence $|uw| \geq \frac{q}{2}$.

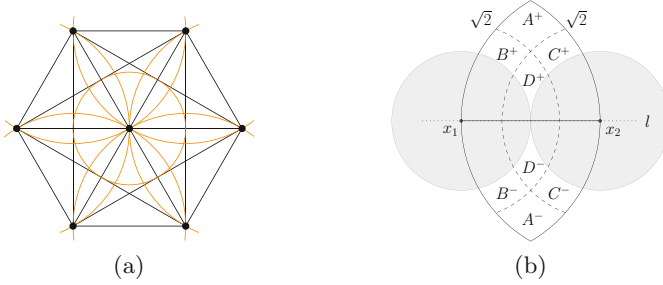


Fig. 3. (a) Empty-ply drawing K_7 ; note that there are edges drawn on top of each other. (b) Partition of the region where the vertices of K_8 can be placed.

From the above reasoning it follows that q should satisfy the quadratic inequality $\frac{q^2}{4} \leq 1 + q^2 - 2q \cos \alpha$, which yields that either $q \leq \frac{4 \cos \alpha - \sqrt{16 \cos^2 \alpha - 12}}{3}$ or $q \geq \frac{4 \cos \alpha + \sqrt{16 \cos^2 \alpha - 12}}{3}$. This contradicts the fact that $q \in [1, \sqrt{2}]$, because: $4 \cos \frac{2\pi}{13} - \sqrt{16 \cos^2 \frac{2\pi}{13} - 12} \doteq 2.8 < 3$ and $4 \cos \frac{2\pi}{13} + \sqrt{16 \cos^2 \frac{2\pi}{13} - 12} \doteq 4.27 > 4.24 \doteq 3\sqrt{2}$. This concludes the proof of the theorem. \square

Note that $K_{1,24}$ admits an empty-ply drawing with only two different lengths of edges (see Fig. 2b) and so the degree bound provided in Theorem 3 is tight.

4 Graph Classes with and Without Empty-Ply Drawings

4.1 Complete Graphs

Theorem 4. *Graph K_n admits an empty-ply drawing if and only if $n \leq 7$.*

Proof (sketch). For a contradiction, suppose that K_8 has an empty-ply drawing Γ . Let (x_1, x_2) be the longest edge of Γ , w.l.o.g. having length 2; assume that x_1 and x_2 lie on an horizontal line l . Since (x_1, x_2) is the longest edge, the remaining six vertices lie in the intersection of two disks centered at x_1 and x_2 , respectively, with radius 2; also, by Lemma 2, they lie outside the two disks centered at x_1 and x_2 with radius 1; see Fig. 3b. This defines two closed regions in which these vertices lie: one above l and one below.

Using two circles centered in x_1 and x_2 with radius $\sqrt{2}$, we partition each of these two regions into four closed subregions, called A^+, B^+, C^+, D^+ and A^-, B^-, C^-, D^- , where the apex $^+$ or $^-$ indicates the region above or below l , respectively. Namely, any point in the interior of $A^+ \cup A^-$ (of $D^+ \cup D^-$) has distance larger (smaller) than $\sqrt{2}$ from both x_1 and x_2 ; while any point in the interior of $B^+ \cup B^-$ (of $C^+ \cup C^-$) has distance smaller (larger) than $\sqrt{2}$ from x_1 and distance larger (smaller) than $\sqrt{2}$ from x_2 .

We show that any placement of the six remaining vertices in these regions leads to a contradiction. We denote by $|X^y|$, with $X \in \{A, B, C, D\}$ and $y \in \{+, -\}$,

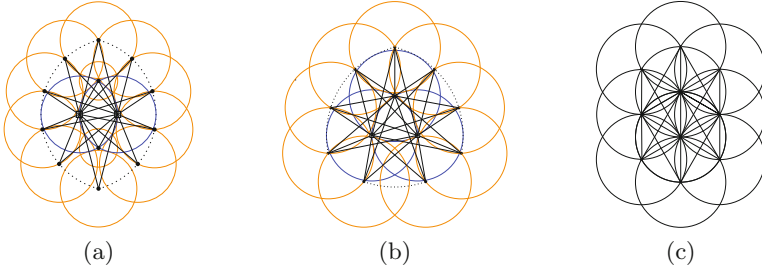


Fig. 4. Empty-ply drawing of (a) $K_{2,12}$, (b) $K_{3,9}$, and (c) $K_{4,6}$. Note that the drawing of $K_{4,6}$ has edges drawn on top of each other.

the number of vertices in X^y . First note that each region can contain at most one vertex, except for D^+ and D^- , which may contain two vertices. In fact, if we place any vertex w in a region X^y , with $X \in \{A, B, C\}$ and $y \in \{+, -\}$, then the ply-disk D_w of w (defined, at least by the distance to x_1, x_2) covers the entire region X^y . Regions D^+ and D^- , on the other hand, have area with height 1 and width 0.5. Let $w \in D^+$ be the point at distance $\sqrt{2}$ from both x_1 and x_2 and D_w be its ply-disk. Then, set $D^+ \setminus D_w$ defines an area with diameter at most $\frac{1}{3}$ and it is not sufficient to place more than one vertex, since the ply disks would have at least a radius 0.5.

Combining the placement of the vertices in different regions, we can use similar arguments to prove that $|D^+ \cup D^-| \leq 3$ and $|A^+ \cup A^-| \leq 1$. Also, if $|A^+| = 1$ (resp. $|A^-| = 1$), then $|D^-| \leq 1$ (resp. $|D^+| \leq 1$). Thus, if $|A^+ \cup A^-| = 1$ then $|D^+ \cup D^-| \leq 2$. Also, if $|A^+| = 1$ (resp. $|A^-| = 1$) and $|B^-| = 1$ (resp. $|B^+| = 1$) then either $|B^+| = 0$ or $|C^+| = 0$ (resp. $|B^-| = 0$ or $|C^-| = 0$), i.e., $|B^+ \cup C^+| \leq 1$ (resp. $|B^- \cup C^-| \leq 1$). By symmetry, if $|A^+| = 1$ (resp. $|A^-| = 1$) and $|C^-| = 1$ (resp. $|C^+| = 1$) then either $|B^+| = 0$ or $|C^+| = 0$ (resp. $|B^-| = 0$ or $|C^-| = 0$), i.e., $|B^+ \cup C^+| \leq 1$ (resp. $|B^- \cup C^-| \leq 1$). Hence, if $|A^+ \cup A^-| = 1$, the other regions cannot contain 5 vertices.

The final case where $|A^+ \cup A^-| = 0$ directly implies the claim for K_9 . To prove this for K_8 we can see that if $|B^-| = 1$ and $|C^+| = 1$ (resp. $|B^+| = 1$ and $|C^-| = 1$), then $|D^+ \cup D^-| \leq 1$, which again leads to a contradiction. To conclude the proof, we present an empty-ply drawing for K_7 in Fig. 3a. We strongly believe that this drawing is unique.

4.2 Complete Bipartite Graphs

We now consider complete bipartite graphs. For proof-by-picture of the next theorem see Fig. 2b and Figs. 4a–c.

Theorem 5. *Graphs $K_{1,24}$, $K_{2,12}$, $K_{3,9}$, and $K_{4,6}$ admit empty-ply drawings.*

Note that Theorem 3 implies that $K_{1,25}$ does not admit any empty-ply drawing, and hence this is true for any complete bipartite graph $K_{n,m}$ with n or m

greater than 24. This leaves a wide open gap between the upper bounds on the values of n and m , and the lower bounds from Theorem 5.

For $K_{2,m}$, we give a negative result for $m \geq 15$ in the following theorem based on arguments similar to those in Theorem 4.

Theorem 6. *Graph $K_{2,m}$ with $m \geq 15$ does not admit any empty-ply drawing.*

4.3 Trees of Bounded Degree

A d -ary tree T with k levels is a rooted tree where all vertices at distance less than k from the root have at most d children and the remaining ones are leaves. If all the non-leaf vertices have exactly d children, we say that T is *complete*. Any tree with maximum degree Δ is a subtree of a $(\Delta - 1)$ -ary tree.

Note that binary trees admit empty-ply drawings, as the drawings with ply 2 constructed by the algorithm in [10] are empty-ply drawings. Applying Corollary 1 to the class of complete 10-ary trees (which do not admit drawing with constant ply [1]) shows that they do not admit empty-ply drawing. But we can prove something stronger.

Theorem 7. *For sufficiently large k , the complete 4-ary tree T_k with k levels admits no empty-ply drawing.*

Proof. Assume without loss of generality that k is even and that T_k has an empty-ply drawing Γ where the ply-disk of the root v_0 has unit radius. We announce that for simplicity the following estimates are not stated in the tightest form. We will make use of the following consequences of Lemmas 2 and 3:

Claim (A). If a ply-disk of a vertex u in Γ has radius at least 2^i , then all the leaves of the subtree rooted at u have radii at least 2^{2i-k} .

Proof. Since $r_u \geq 2^i$, the distance between u and the root is greater than i by Lemma 3. Thus the path from u to its leaves has length at most $k - i$. \square

Claim (B). If v is a leaf whose ply-disk has radius $r_v \in (2^{2i-k}, 2^{2i-k+2}]$, with $i \in \{0, k - 1\}$, then its Euclidean distance from the root is $|v_0v| \leq 2^{i+2}$.

Proof. Let $v_0, v_1, \dots, v_k = v$ be the path from the root v_0 to v_k in T_k . Since $r_{v_0} = 1$, edge (v_0, v_1) has length at most 2. Also, by Lemma 2, the lengths of the edges can grow at most by a factor 2 along the path; hence, $|v_{j-1}v_j| \leq 2^j$ for $j \in \{1, \dots, i\}$. If we traverse the path in the opposite direction from v_k , whose ply-disk has radius at most 2^{2i-k+2} , we get analogously that $|v_{k-j+1}v_{k-j}| \leq 2^{j+2i-k+2}$ for $j \in \{1, \dots, k - i\}$.

The total distance is thus bounded by $|v_0v_k| \leq |v_0v_1| + |v_1v_2| + \dots + |v_{k-1}v_k| = \sum_{j=1}^i |v_{j-1}v_j| + \sum_{j=1}^{k-i} |v_{k-j+1}v_{k-j}| \leq \sum_{j=1}^i 2^j + \sum_{j=1}^{k-i} 2^{j+2i-k+2} = 2^{i+1} - 2 + 2^{i+1} - 2^{3+2i-k} \leq 2^{i+2}$, and the statement follows. \square

We now distribute the 4^k leaves to k sets L_0, \dots, L_{k-1} (all logarithms binary):

- (a) if $i \geq 3 \log k$ then $L_i = \{v : r_v \in (2^{2i-k}, 2^{2i-k+2}]\}$
- (b) if $i < 3 \log k$ then $L_i = \{v : r_v \leq 2^{6 \log k - k}$ and whose largest predecessor u has radius $r_u \in [2^i, 2^{i+1}]\}$

In the first case, the radii of the leaves in L_i are sufficient to obtain a good bound on enclosing area of the disks in L_i . In the other case, the radius on the enclosing disk for L_i mostly depends on the presence of predecessors that are larger than the root disk.

Some of the sets are empty by the definition, but it is irrelevant to our further deductions. By pigeonhole principle, either some L_i , $i \geq 3 \log k$ satisfies $|L_i| \geq \frac{4^k}{2k}$ or some L_i , $i < 3 \log k$ satisfies $|L_i| \geq \frac{4^k}{6 \log k}$, since $\frac{k+1-3 \log k}{2k} + \frac{3 \log k}{6 \log k} \leq 1$ when $k \geq \sqrt[3]{2}$.

The rough idea behind the distinction of these two cases is that in case a), when the diameters of leaves are sufficiently large, it suffices to consider twice smaller proportion than the uniform pigeonhole principle would use and show that the total area of ply-disks corresponding to leaves of L_i is still too large for an empty-ply drawing Γ . In case b) we use a slightly more elaborate argument considering also the area of the predecessors of the vertices in L_i .

Case (a). Assume that for some $i \geq 3 \log k$ it holds that $|L_i| \geq \frac{4^k}{2k}$. The total area occupied by the disks in L_i is at least $\frac{4^k}{2k} \pi 4^{2i-k} = \frac{8^i \pi}{2k}$. By Claim (B), for every $v \in L_i$ it holds that $|v_0 v| \leq 2^{i+1}$, hence all ply-disks of L_i must be contained in a disk centered at the root of radius $2^{i+1} + 2^{2i-k+2} \leq 5 \cdot 2^i$, since for $i \in \{0, \dots, k\} : i > 2i - k$. In particular this disk has area at most $25\pi 4^i$.

In order to apply Lemma 1, it suffices to choose k large enough such that $\frac{8^i \pi}{2k} > 4 \cdot 25\pi 4^i$ for all $i \geq 3 \log k$, i.e., $k > \sqrt[5]{200} \doteq 2.9$.

Case (b). Assume that for some $i < 3 \log k$ it holds $|L_i| \geq \frac{4^k}{6 \log k}$. Any $v \in L_i$ has radius smaller than $2^{6 \log k - k}$, as otherwise we would be in case a). To obtain the maximum distance between v and the root v_0 we argue that the first $3 \log k$ disks along the path from v_0 to v may have radius at most 2^{i+1} . Analogously as in the proof of Claim (B), the j -th predecessor of v has radius at most $2^{6 \log k - k + j}$. An upper bound of $|v_0 v| \leq 2^{i+2}(3 \log k + 1)$ is obtained by summing up.

We now consider the subtree T' of T_k induced by the vertices of L_i and all their predecessors. Note that the drawing of the entire tree T' shall be contained within a disk of radius $2^{i+2}(3 \log k + 1) + 2^{3 \log k - k}$, i.e., in area at most $4^{i+3} \pi$. On the other hand, by Claim (A), each of the leaves has radius at least 2^{2i-k} . Thus, their total area is at least $\frac{4^k}{6 \log k} 4^{2i-k} \pi = \frac{8^i \pi}{6 \log k}$.

The number of parents of disks in L_i is at least $\frac{4^{k-1}}{6 \log k}$, each of radius at least 2^{i-1} , hence they occupy area also bounded from below by $\frac{8^i \pi}{6 \log k}$. Thus, all leaves in L_i and all their $k - i$ predecessors occupy space at least $\frac{8^i \pi}{6 \log k} (k - i) \geq \frac{8^i \pi k}{12 \log k}$. Again, to apply Lemma 1, it suffices to choose k large enough such that

$\frac{8^i \pi k}{12 \log k} > 4 \cdot 64\pi 4^i$ for all non-negative $i < 3 \log k$ (in particular for $i = 0$). A straightforward calculation verifies that the inequality holds e.g., for $k \geq 2^{16}$.

For $k = 2^{16}$ one of the two cases applies, which concludes the proof. \square

Theorem 7 leaves open the question for 3-ary trees. We remark that the algorithm for binary trees [10] adopts a common drawing style: the orthogonal one with a shrinking factor of 1/2; see also [11]. We prove that this technique fails for 3-ary trees, for any shrinking factor in $(0, 1)$.

Theorem 8. *Rooted ternary trees do not admit empty-ply drawings constructed in orthogonal fashion with shrink factor q for any $q \in (0, 1)$, i.e., when the distance from a vertex to its children is q times the distance to its parent.*

4.4 Graph Squares

The *square* of a graph G is the graph obtained from G by adding an edge between each vertex and the neighbors of its neighbors.

Theorem 9. *Let G^2 be the square of a graph G . If G admits a drawing with ply 1, then G^2 admits an empty-ply drawing. Also, if G is a subgraph of a triangular tiling, then G^2 admits an empty-ply drawing with ply at most 4.*

Proof. Let Γ be a straight-line drawing of G with ply 1. As proved in [10], all the edges of G have the same length, say 1, in Γ , and every two non-adjacent vertices are at distance at least 1 from each other. Hence, adding the edges of $G^2 \setminus G$ to Γ produces a drawing Γ^2 of G^2 in which each edge has length at most 2. This implies that every ply-disk has radius at most 1 in Γ^2 , and thus Γ^2 is an empty-ply drawing. Note that Γ^2 may contain edge overlaps.

For the second part of the statement, recall that if G is a subgraph of a triangular tiling, then it admits a drawing Γ in which all edges have the same length and all the angles are multiples of $\frac{\pi}{3}$. Hence, Γ has ply 1. Also the drawing Γ^2 obtained by adding the additional edges of $G^2 \setminus G$ to Γ is an empty-ply drawing. In this case, however, we can also prove that the ply of Γ^2 is at most 4; recall that an upper bound of 5 to the ply of Γ^2 is already implied by Corollary 1.

W.l.o.g. let the triangular tiling be of unit edge length. Consider the open disk of unit radius, which is centered at an arbitrary point p on the plane. If p is not a vertex of the triangular tiling, at most four vertices of the triangular tiling may fall in this disk. In the case where p is a vertex of the triangular tiling, no other vertex of the tiling falls in the disk, but only on its boundary. Thus, any point p can be internal to at most four ply-disks of the tiling vertices. \square

5 Ply and Vertex-Ply of Planar Drawings

In the original paper on the ply number it was observed that considering only plane graph drawings may prevent finding low ply non-plane drawings [10]. In particular, for the class of *nested-triangles* graphs the “most natural” planar

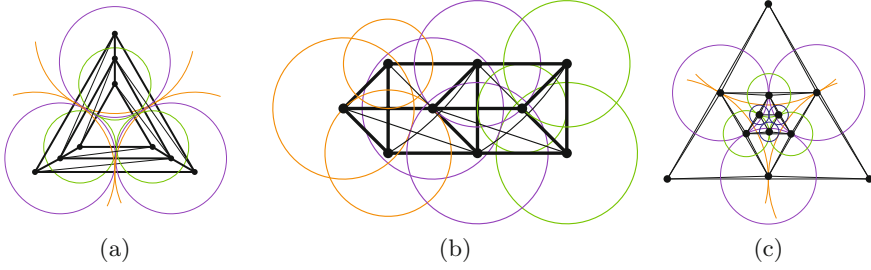


Fig. 5. Nested triangles graph: (a) “The most natural” drawing. (b) A non-planar drawing with ply 5. (c) A planar drawing with ply 4. The disks of three vertices at the same level do not properly overlap, and disks at levels i and $i + 3$ do not overlap.

drawing has ply $\Omega(n)$ (see Fig. 5a), while there exist non-planar drawings (with edge overlaps) with ply 5 (see Fig. 5b). Note that however a “less natural” planar drawing with ply 4 can always be constructed; see Fig. 5c.

We strengthen this observation by providing a planar 3-tree G admitting a non-planar drawing (with only 3 crossings) with ply 5, such that *any* planar drawing of G has ply $\Omega(n)$; the same linear lower bound holds even for vertex-ply when the outer face is fixed. Recall that a *planar 3-tree* can be constructed, starting from a 3-cycle, by repeatedly adding a vertex inside a triangular face and connecting it to all three vertices of this face.

Our result also gives a negative answer to an open question posed in [10] on whether there exists a relationship between the number of crossings and the ply number of a drawing. Our example shows that one can reduce the ply number from $\Omega(n)$ to $O(1)$, by introducing only $O(1)$ crossings.

Theorem 10. *There exists an n -vertex planar 3-tree G such that any planar drawing of G with a fixed outer face has vertex-ply $\Theta(n)$, and hence ply $\Theta(n)$, while G admits a drawing with ply 5 and vertex-ply 4 with three edge crossings.*

Proof. Graph G has three vertices v_1, v_2 , and v_3 on the outer face, and a vertex u that is connected to all of v_1, v_2 , and v_3 . Refer to Fig. 6a. In addition, it contains three paths $x_1, \dots, x_m, y_1, \dots, y_m$, and z_1, \dots, z_m , each on $m = \frac{n-4}{3}$ vertices. The edge set further contains edges $(u, x_1), (u, y_1), (u, z_1)$ and also $(x_i, v_1), (x_i, v_2), (y_i, v_2), (y_i, v_3), (z_i, v_1), (z_i, v_3)$ for each $i \in \{1, \dots, m\}$.

Consider any planar drawing Γ of G . Suppose, w.l.o.g., that (v_1, v_2) is of unit length and that it is the longest edge in Γ among the three edges incident to the outer face, that is, $|v_2v_3|, |v_1v_3| \leq 1$. Since vertex u lies inside the triangle $v_1v_2v_3$, we have $|uv_1|, |uv_2| < 1$. Hence, it is possible to cover the whole region of the plane delimited by triangle uv_1v_2 with a set of 28 disks, each having radius $\frac{1}{8}$, as illustrated in Fig. 6b. Thus, at least one disk D out of these 28 contains in its interior at least $\frac{m}{28} = \frac{n-4}{84}$ vertices out of x_1, \dots, x_m .

Consider any vertex $x_i \in D$. Since x_i is connected to both v_1 and v_2 , the longest of its incident edges has length at least $\frac{1}{2}$, and hence the radius of the ply-disk of x_i is at least $\frac{1}{4}$. Hence the ply-disk of x_i entirely contains the disk D

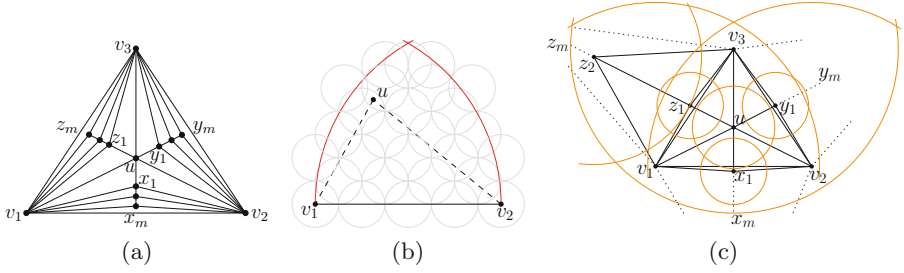


Fig. 6. (a) The planar 3-tree G in the proof of Theorem 10. (b) A set of 28 disks of radius $\frac{1}{8}$ covering the whole region delimited by triangle uv_1v_2 when $|v_1v_2| = 1 > |uv_1|, |uv_2|$. (c) A non-planar drawing of G with ply 5 and vertex-ply 4.

in its interior, and thus it contains all the vertices inside it. Since this is true for all the $\frac{n-4}{84}$ vertices inside D , the first part of the statement follows.

A non-planar drawing of G with ply 5 and vertex-ply 4 is depicted in Fig. 6c. Here vertices v_1, v_2 and v_3 form an equilateral triangle with barycenter u . Vertices x_1, \dots, x_m are arranged along the axis of the segment v_1v_2 at distances growing exponentially by a factor of 2, analogously for vertices y_1, \dots, y_m and z_1, \dots, z_m . The disk D_u overlaps with D_{x_1}, D_{y_1} , and D_{z_1} , without enclosing these vertices. The drawing of the subset of vertices $\{u, x_1, y_1, z_1\}$ is empty-ply and of ply 2. After considering the remaining vertices, the disks of v_1, v_2, v_3 may contain all of them in their interior. Thus we obtain ply 5 and vertex-ply 4.

6 Conclusions and Future Work

We defined and studied the vertex-ply of a straight-line drawing, paying particular attention to the special case of empty-ply drawings, whose vertex-ply is 1. We conclude with several natural open problems.

1. We know that binary trees admit empty-ply drawings [10] and that 4-ary trees do not (Theorem 7). What about 3-ary trees? Note that Theorem 8 rules out a large class of possible drawings (orthogonal and shrinking).
2. Another way of generalizing binary trees is to maintain the degree restriction, leading to the question: do (planar) max-degree-3 graphs admit empty-ply drawings?
3. In Theorem 9 we proved that the square G^2 of a graph G with ply 1 admits an empty-ply drawing, which has ply at most 5 by Corollary 1. On the other hand, if G is a subgraph of a triangular tiling, then the empty-ply drawing of G^2 has ply at most 4. Does the square of every graph with ply 1 admit an (empty-ply) drawing with ply 4? Note that there are ply 1 graphs that are not subgraphs of a triangular tiling.
4. Looking at empty-ply drawings from the proximity perspective, it is natural to consider the generalization in which ply-disks do not need to be empty, but can contain at most k vertices. We call a drawing with this property

- a *k*-empty-ply drawing, in compliance with the definition of *k*-Gabriel and *k*-relative-neighborhood drawings [17]. With the argument of Theorem 10 there exist *n*-vertex graphs whose any planar drawing is $\Omega(n)$ -empty-ply.
5. In Theorem 4 we proved a tight bound of 7 on the size of complete graphs admitting empty-ply drawings. For complete bipartite graphs $K_{n,m}$, we have a tight bound of $m = 24$, for $n = 1$, and an almost tight bound of $12 \leq m \leq 14$, for $n = 2$, with larger gaps between the bounds for larger values of *n*.

Acknowledgments. This work began at the 2015 HOMONOLO meeting. We gratefully thank M. Bekos, T. Bruckdorfer, G. Liotta, M. Saumell, and A. Symvonis for great discussions on the topic. Research was partially supported by project CE-ITI P202/12/G061 of GAČR (J.F., J.K., P.V.), by project SVV–2017–260452 (J.H.), and by DFG grant Ka812/17-1 (P.A., N.H., M.K.).

References

1. Angelini, P., Bekos, M.A., Bruckdorfer, T., Hančl, J., Kaufmann, M., Kobourov, S., Symvonis, A., Valtr, P.: Low ply drawings of trees. In: Hu, Y., Nöllenburg, M. (eds.) GD 2016. LNCS, vol. 9801, pp. 236–248. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_19
2. Angelini, P., Chaplick, S., De Luca, F., Fiala, J., Hančl Jr., J., Heinsohn, N., Kaufmann, M., Kobourov, S., Kratochvíl, J., Valtr, P.: On vertex- and empty-ply proximity drawings. ArXiv e-prints [1708.09233](https://arxiv.org/abs/1708.09233) (2017)
3. Breu, H., Kirkpatrick, D.G.: Unit disk graph recognition is NP-hard. *Comput. Geom.* **9**(12), 3–24 (1998)
4. Bruckdorfer, T., Cornelsen, S., Gutwenger, C., Kaufmann, M., Montecchiani, F., Nöllenburg, M., Wolff, A.: Progress on partial edge drawings. In: Didimo, W., Patrignani, M. (eds.) GD 2012. LNCS, vol. 7704, pp. 67–78. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36763-2_7
5. Bruckdorfer, T., Kaufmann, M.: Mad at edge crossings? Break the edges!. In: Kranakis, E., Krizanc, D., Luccio, F. (eds.) FUN 2012. LNCS, vol. 7288, pp. 40–50. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30347-0_7
6. Bruckdorfer, T., Kaufmann, M., Lauer, A.: A practical approach for 1/4-SHPEDs. In: 6th International Conference on Information, Intelligence, Systems and Applications (IISA), pp. 1–6. IEEE (2015)
7. De Luca, F., Di Giacomo, E., Didimo, W., Kobourov, S., Liotta, G.: An experimental study on the ply number of straight-line drawings. In: Poon, S.-H., Rahman, M.S., Yen, H.-C. (eds.) WALCOM 2017. LNCS, vol. 10167, pp. 135–148. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-53925-6_11
8. Delaunay, B.: Sur la sphère vide. a la mémoire de Georges Voronoi. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et naturelles*, vol. 6, pp. 793–800 (1934)
9. Di Battista, G., Liotta, G., Whitesides, S.: The strength of weak proximity. *J. Discrete Algorithms* **4**(3), 384–400 (2006)
10. Di Giacomo, E., Didimo, W., Hong, S., Kaufmann, M., Kobourov, S.G., Liotta, G., Misue, K., Symvonis, A., Yen, H.: Low ply graph drawing. In: 6th International Conference on Information, Intelligence, Systems and Applications (IISA), pp. 1–6. IEEE (2015)

11. Eiglsperger, M., Fekete, S.P., Klau, G.W.: Orthogonal graph drawing. In: Kaufmann, M., Wagner, D. (eds.) *Drawing Graphs*. LNCS, vol. 2025, pp. 121–171. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44969-8_6
12. Eppstein, D., Goodrich, M.T.: Studying (non-planar) road networks through an algorithmic lens. In: *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 1–10. ACM (2008)
13. Formann, M., Hagerup, T., Haralambides, J., Kaufmann, M., Thomson Leighton, F., Symvonis, A., Welzl, E., Woeginger, G.J.: Drawing graphs in the plane with high resolution. In: *FOCS*, pp. 86–95. IEEE Computer Society (1990)
14. Fruchterman, T.M.J., Reingold, E.M.: Graph drawing by force-directed placement. *Software Pract. Exp.* **21**(11), 1129–1164 (1991)
15. Gabriel, K.R., Sokal, R.R.: A new statistical approach to geographic variation analysis. *Syst. Biol.* **18**(3), 259–278 (1969)
16. Heinsohn, N., Kaufmann, M.: An interactive tool to explore and improve the ply number of drawings. In: *Proceedings of the 25th International Symposium on Graph Drawing and Network Visualization (GD 2017)* (2017, to appear)
17. Liotta, G.: Proximity drawings. In: Tamassia, R. (ed.) *Handbook on Graph Drawing and Visualization*, pp. 115–154. Chapman and Hall/CRC (2013)
18. Malitz, S.M., Papakostas, A.: On the angular resolution of planar graphs. In: Kosaraju, S.R., Fellows, M., Wigderson, A., Ellis, J.A. (eds.) *ACM Symposium on Theory of Computing*, pp. 527–538. ACM (1992)
19. Preparata, F., Shamos, M.: *Computational Geometry: An Introduction*. Texts and Monographs in Computer Science. Springer, New York (1985)
20. Toussaint, G.T.: The relative neighbourhood graph of a finite planar set. *Pattern Recogn.* **12**(4), 261–268 (1980)

An Interactive Tool to Explore and Improve the Ply Number of Drawings

Niklas Heinsohn^(✉) and Michael Kaufmann

Wilhelm-Schickhard-Institut für Informatik, Universität Tübingen,
Tübingen, Germany
{heinsohn,mk}@informatik.uni-tuebingen.de

Abstract. Given a straight-line drawing Γ of a graph $G = (V, E)$, for every vertex v the ply disk D_v is defined as a disk centered at v where the radius of the disk is half the length of the longest edge incident to v . The ply number of a given drawing is defined as the maximum number of overlapping disks at some point in \mathbb{R}^2 . Here we present a tool to explore and evaluate the ply number for graphs with instant visual feedback for the user. We evaluate our methods in comparison to an existing ply computation by De Luca et al. [WALCOM'17]. We are able to reduce the computation time from seconds to milliseconds for given drawings and thereby contribute to further research on the ply topic by providing an efficient tool to examine graphs extensively by user interaction as well as some automatic features to reduce the ply number.

1 Introduction

Graphs are the common mathematical model to represent relationships between objects and occur in a huge variety of applications and disciplines. To make the data stored in a graph accessible for humans, we need a graphical representation which usually involves a drawing of the underlying graph. There exist several schemes to draw graphs [8, 16]. In this work we will focus on straight-line drawings. Several aesthetic criteria on straight-line drawings have been defined to capture the user requirement for a better understanding of the data (e.g. edge crossings or angular resolution [3]).

Recently a new parameter called ply number was introduced as a quality metric for graph layouts [9]. Given a straight-line drawing Γ of a graph $G = (V, E)$, for every vertex v the ply disk D_v is defined as a disk centered at v , where the radius of the disk is half the length of the longest edge incident to v . The ply number of Γ is the maximal number of overlapping disks at any point in \mathbb{R}^2 . Theoretical results have been obtained on the ply number of graphs [1, 9] and there exist many real world graphs admitting natural drawings with low ply number [10]. One common approach to draw such graphs are force-directed algorithms [15], whose drawings are known to be aesthetically pleasing. A recent study evaluated the correlation between the ply number of drawings produced by force-directed algorithms and other known metrics defined on these algorithms [7].

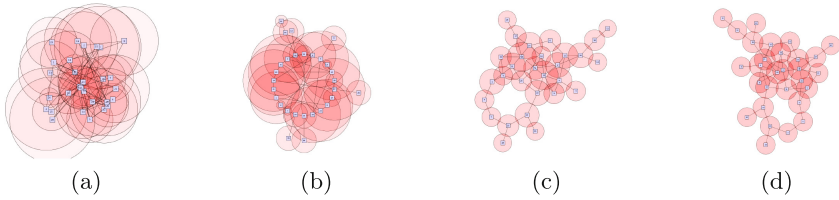


Fig. 1. A graph with 30 vertices and 40 edges is drawn by our tool in different layouts: (a) randomly placed vertices with ply 15, (b) circular layout with ply 7, (c) organic layout with ply 4, (d) the lowest known ply number of 3 for this graph.

There exist many tools and layout algorithms for graph drawing provided e.g. by OGDF [6] or the yFiles library [19]. The identification of properties as well as the development of new strategies to optimize parameters of drawings involve frequent examination of graph drawings. In this paper we present a tool which allows investigation of graphs according to its ply number. We present a fast algorithm to compute the ply number for a given drawing based on a plane-sweep algorithm which is known to be a powerful technique in computational geometry. Furthermore we provide methods to modify the drawing to reduce the ply number interactively as well as automatically. We confirm the value of our tool by providing experimental results on the computation in terms of accuracy and speed as well as the results on our ply minimization approaches.

A first prototype of the basic plane-sweep algorithm has been implemented in [4]. We reimplemented the algorithm, improved it and added new features. The experimental study of De Luca et al. [7] set a benchmark on the computation of the ply number for a given drawing. The authors evaluated several layouts by force directed algorithms according to the ply number. To make our comparison possible, the authors kindly provided some of their data. Both implementations of the ply computations are based on the Apfloat library [17] which allows calculations on high precision levels at the cost of time.

2 Functionality

Our tool allows the investigation of a graph regarding its ply number. As basic functionality, the tool is equipped with some graph layout algorithms, namely organic, circular and randomized (provided by yFiles library [19]), as presented in Fig. 1, and allows for interactive manipulation of the drawing, such as moving vertices. Basic file formats are graphml [5] and gml [14] where graphml provides structural information on the graph and gml provides a drawing.

Furthermore, we provide a test if a given drawing is empty-ply, where no vertex is contained in any other vertex's disk. With our tool we identified that the complete graph $K_{4,6}$ admits an empty-ply drawing whereas this was previously known only for $K_{4,4}$ [2]. Our implementation is able to compute the ply number for the drawing during runtime, meaning while the drawing is modified, for example by layout algorithms or user interaction.

Live feedback of the ply number on interactive graph manipulations by users is another feature of our tool. We mark regions where the maximal ply number occurred. The user can choose between different layouts as start configurations for the graph and is able to improve the ply values automatically by a spring embedder or by manually manipulating the positions of the vertices accordingly.

3 Ply Computation Algorithm

To compute the ply number of a drawing Γ we have to compute the intersections of the ply disks. Clearly the ply number is at most linear in the number of vertices. In the following, we introduce one major issue in computing ply numbers and present our fast plane-sweep algorithm to compute the ply number.

3.1 Precision Problems

Naturally thinking of an easy case to start with are graphs that admit a drawing with a ply number of 1. This case is easy to describe and points out the difficulty of computing the ply number: A graph that admits a drawing with ply number 1 has no overlapping ply disks and can be drawn such that every edge has equal length l and any two vertices are at a distance of at least l . Suppose that there exists a drawing Γ and a vertex v with different edge lengths $|(u, v)| = l_1$ and $|(w, v)| = l_2$ and let w.l.o.g $l_1 > l_2$. $\frac{l_2}{2}$ is a lower bound on the ply disk D_w and since $\frac{l_1}{2} + \frac{l_2}{2} > l_2$ the ply disks D_v and D_w intersect and the ply number of Γ is ≥ 2 . Furthermore, since the radius for any ply disk D_v is $\frac{l}{2}$, the distance $\text{dist}(v, w)$ between any two vertices has to be $\geq l$.

The complete graph K_3 admits a drawing with ply number 1, since the vertices can be placed on an equilateral triangle. Computing a drawing of K_3 with the vertices u, v , and w , some coordinates must be irrational, since otherwise the condition $\text{dist}(u, v) = \text{dist}(u, w) = \text{dist}(v, w)$ is violated. As a consequence the computer would need an infinite precision to represent a drawing with ply number 1. Furthermore the calculation of coordinates for intersection points of circles involves precise arithmetic and is likely to result in irrational coordinates.

In previous applications [4, 7] this problem was tackled by increasing the precision using the `Apfloat` library. This allows calculation on up to 1000 digit decimal precision. On the downside these arithmetics require high computational effort. In the following we present an alternative approach using the primitive type `double` reducing the computational effort. Later, we evaluate this decision by experiments regarding the precision of the outcome in terms of events and time spent computing the ply number.

3.2 Plane-Sweep Algorithm

A plane-sweep algorithm describes a powerful technique in computational geometry. Given a two-dimensional Euclidean space, a conceptual line which represents the actual state of computation sweeps the Euclidean space scanning for events

from left to right. Given a drawing Γ of a graph $G = (V, E)$, we can easily compute the set of ply disks $D = \{D_v | v \in V\}$. Every disk D_v is associated with the vertex v at position (x_v, y_v) and radius r_v . At every x-coordinate of the setting there exists a highest ply value. Note that the ply number of the graph is the maximum over all ply values. The ply value can change whenever a disk starts at $(x_v - r_v, y_v)$, ends at $(x_v + r_v, y_v)$ or if there is an intersection of two disks. For our purpose these coordinates are called events.

To describe the vertical structure with opening and closing disks, we represent the disks as bottom and top halfcircles, respectively to the center. At any x-coordinate between two consecutive events the order of opening and closing halfcircles on a vertical line in a ply drawing and thereby the maximal ply value is fix. For every halfcircle we associate and store the ply value.

Whenever we meet a leftmost coordinate $x_v - r_v$ of a disk D_v , we introduce two halfcircles (h_v^t and h_v^b) and add them in the vertical structure. We set the ply values according to the neighbored halfcircles in this case. Whenever we meet a rightmost coordinate $x_v + r_v$ of a disk D_v , we remove both corresponding halfcircles from the vertical structure. If there exist halfcircles between our removed ones, we decrease the ply of these.

Note that two disks might intersect if and only if any two corresponding halfcircles occur next to each other in the vertical structure. Furthermore, any two disks D_u and D_v can intersect at most twice. To keep the computational effort minimal the intersection of disks is calculated the first time any two halfcircles appear next to each other in the vertical structure. Finally an intersection-event swaps the two affected halfcircles h_u and h_v , the ply is updated due to a case distinction.

The events are stored in a priority queue and are executed by their x-coordinate. In case there exist several events at the same x-coordinate we define the priority in ascending order as end-event, intersection-event, start-event.

The x-coordinates might get slightly inconsistent due to previously mentioned precision errors. This results in events which cannot be handled consistently. One example would be an intersection-event that requires a swap of halfcircles, which are not neighbored in the actual state. Our solution to this scenario is linearly searching for the closest consistent event. This event will be executed and we jump back to the unresolved one. We repeat this until it can be resolved. These events will be tracked as **postponed events**. In the results section we evaluate this delay and describe graphs where this periodically occurs.

4 Experiments

This section is subdivided into three major parts. At first we will present results on the ply number for various graphs, as well as the number of events and the time. Second, we compare our results with the results of De Luca et al. [7]. Third, we present an approach to reduce the ply number by local modification.

We will make use of three datasets. First we take a subset of the Rome graphs [18], which is determined by taking all graphs matching the file name

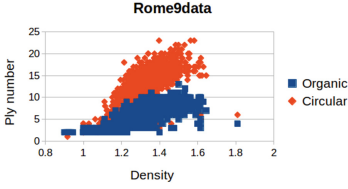


Fig. 2. Ply number for the Rome graphs with various densities.

Table 1. The table presents the average values on **Rome9data**.

Layout	Ply	Time (ms)	Events
Organic	6.3	< 1	546
Circular	12.5	1.1	974
Random	30	3.8	3153

pattern grafo9*. We call the set **Rome9data**. It consists of 1094 sparse graphs with 10 to 100 nodes and a density between 0.9 and 1.8.

The second set contains 100 randomly generated graphs. Every graph consists of 100 vertices and has densities between 1.5 and 40. We refer to this set as **RANDdata**.

The third set will be referred to as **FM3data**. This set of graphs was kindly provided by the authors of the experimental study [7] and each graph was drawn using the fast multipole multilevel method of Hachul and Jünger [12] which is among the most effective force-directed algorithms in the literature [13]. **FM3data** contains caterpillars, planar and general (connected simple graphs, generated with uniform probability distribution) graphs.

4.1 Ply Number for Different Layouts

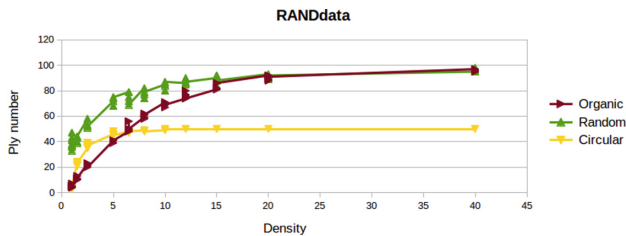
Our tool supports different layouts provided by the yFiles library, namely organic, circular and random layout. We evaluated our algorithm on **Rome9data**. We observe that for sparse graphs the organic layout creates drawings with lower ply than the circular layout, whereas at densities close to one, i.e. tree like, they produce similar ply numbers (see Fig. 2). As a reference the random layout produces the highest ply numbers (see Table 1). We observe a correlation between the ply number, time and events.

The results on **RANDdata** are presented in Table 2. Here, the number of postponed events is noteworthy. While it can be neglected in the organic and random layout, the number of postponed events in the circular layout is consequential and highly increased with the density. The highly symmetric placement of the vertices in the circular layout causes many events to share an x-coordinate. This circumstance, paired with eventually occurring precision errors, influences this number. Since summing up over all postponed events exceeds the total number of executed events, the postponed events highly influence the computation time, as we do a linear search for the next event.

In sparse graphs spring embedding algorithms like the organic layout algorithm produce drawings with low ply numbers, while in dense graphs the drawings have similar ply numbers to a drawing where the vertices are placed randomly. In the dense graphs the circular layout hits the theoretical upper bound of $\frac{|V|}{2}$ as shown in Fig. 3.

Table 2. The average ply numbers for each layout is presented for **RANDdata**, as well as the average computation time and the average number of postponed events.

Density	Layout	Ply	Time (ms)	Events	Postponed
1.5–2.5	Organic	16.2	2.2	2381	0
	Circular	29.3	5.3	4349.5	11.3
	Random	48.2	8.3	7170	0
5–8	Organic	50	12.2	7568	0
	Circular	47.5	15.9	9076.5	109.8
	Random	75	15.2	9510	0
10–15	Organic	76.5	14	9534	0
	Circular	49.7	18	9650.8	3343.6
	Random	86	14.4	9882.5	0
20–40	Organic	93	15.9	10016.4	0
	Circular	50	160.1	9925.6	151232
	Random	94	17.9	10041.4	0

**Fig. 3.** The density of the graphs of **RANDdata** is plotted against the ply number of the drawing. We observe that the organic layout produces low ply numbers for low densities, whereas at higher densities the circular layout outperforms the organic layout. In very dense graphs the organic layout performs evenly as the random layout.

4.2 Comparison on the FM3 Drawing Dataset

We compare the number of events and the average computation time on **FM3data**. The Apfloat decimal precision was set to 20 digits for this comparison. This value was used in the experiments of [7]. We will present the data split by the type of graphs and according to their density.

FM3data contains 50 caterpillars with 250 to 450 vertices. The average number of events and the computation time is presented in Table 3. We observe a huge difference in the total number of events and the computation time. The difference in number of events can be explained due to the difference in handling inconsistent events. The algorithm of [7] introduces a number of redundant events to detect and handle inconsistencies. Our algorithm reduces the computation for the ply number from seconds to milliseconds.

Table 3. The caterpillars of **FM3data**. Each subset with 250 to 450 vertices contains 10 graphs. During all experiments the ply numbers for both implementations were the same. The table presents average values for each set of graphs.

Vertices	Ply	[7]		Our Tool	
		Events	Time (ms)	Events	Time (ms)
250	3.8	2692.4	1328.1	1122.6	3.5
300	4.3	3510.4	1831.9	1430	1.6
350	4.5	3827	1883.7	1602.4	1.9
400	4.6	4564.9	2291.5	1879.3	2.4
450	4.3	5032.8	2581	2110	2.3

Table 4. The planar graphs of **FM3data**. The values show the average results for each subset. Both implementations always computed the same ply numbers.

Density	Ply	[7]		Our Tool	
		Events	Time (ms)	Events	Time (ms)
≤ 1.7	9.6	9878.6	8444.2	3434.6	3.7
> 1.7	11.4	9625.9	9625.9	3609.4	3.8

FM3data also contains planar graphs ranging from 250 to 400 vertices and a density ranging from 1.5 to 2. We split the planar graphs in two density classes. In one set all densities are ≤ 1.7 and in the second set the densities are > 1.7 . The results are presented in Table 4. We observed that the ply number seems to be related to the density rather than the number of vertices. A summary is presented in Fig. 4. As a confirmation of previously made observations we computed the ply of organic and circular layouts. For low densities the circular layout produces higher ply drawings where the organic layout produces similar ply numbers as the FM3 algorithm. On average the drawings generated by FM3 have slightly higher ply than the organic layout. The average ply number in the organic layout is 9 for graphs with density ≤ 1.7 and 9.7 for higher density. Again, note the difference in number of events and computation time keeping results equal.

The remaining subset of **FM3data** consists of general graphs with 250 to 450 vertices and the densities 1.5 and 2.5. In 96 of 100 graphs both implementations computed the same ply number, whereas in 3 graphs we see a difference of 1. In one specific graph, namely `General_400_2.5_5_d.0_FMMM_drawing.gml`, the ply number differs by 5. In these graphs we can detect a high number of postponed events. Furthermore, our algorithm underestimates the ply number in all cases.

To conclude this section we present the interesting result on different layouts on the third subset of **FM3data** presented in Fig. 5. Note that the ply numbers on FM3 and organic layout are very similar. The stairs in the plot indicate the jump between the densities from 1.5 to 2.5 for each set of graphs.

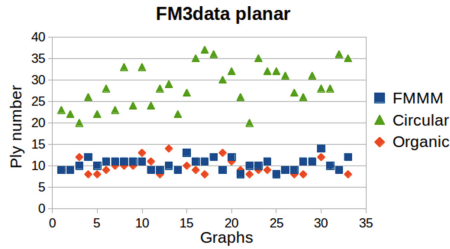


Fig. 4. For each planar graph of **FM3data** the computed ply numbers for each of the three layouts is plotted. Note that the organic and the FM3 drawings have similar ply numbers, while the circular layout produces higher ply numbers on low density graphs.

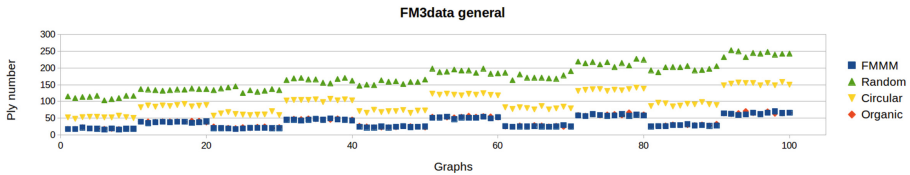


Fig. 5. The set of general graphs can be subdivided in 5 subsets consisting of graphs with 200, 250, ... , 450 vertices. Each subset can be divided in 10 graphs with density 1.5 and 10 graphs with density 2.5. The figure indicates that the **FM3** algorithm and the organic layout produce similar drawings regarding the ply number.

4.3 Ply Minimization

In this part we will present some strategies to create drawings with low ply. We evaluated our strategies on **FM3data** and **Rome9data**. Our first strategy is based on an obvious upper bound of $\frac{|V|}{2}$ on the ply number of any graph $G = (V, E)$, which is obtained by placing the vertices regularly on a circle C . For every disk there exist a unique disk on the opposite side of the center which is not overlapping. Therefore at most half of the disks can contribute to the ply number. We use this observation and apply the circular layout, whenever the actual layout has ply number larger than $\frac{|V|}{2}$.

4.4 Strategies

To achieve a low ply drawing for a given graph we introduce a workflow, which is directly accessible in our tool. We start with the organic layout since it has presented itself to produce drawings with low ply number on sparse graphs. Examining these drawings with our tool, we observed that there often exist very few regions with the maximal ply number, which often can be reduced by moving a few vertices locally. From these observations we adjusted a new spring embedder based on Fruchterman and Reingold [11], similar as suggested in [7], and tuned the parameters to produce drawings with less ply.

4.5 Results

At first we present the advantages of our methods on **Rome9data** in comparison to organic layouts. On average the ply number of 6.3 of the organic layout was improved to 5.1 in the modified setting. This result is presented in Fig. 6.

In the experimental study of the ply number [7] one of the results was the strength of the FM3 algorithm to produce low ply drawings. We compare the ply numbers for **FM3data** to the findings of our ply minimization workflow. Like in the previous chapters we will present the results separately. For all computations of the ply numbers we took our implementation for consistency.

On the caterpillars the average ply number for the **FM3data** is 4.3, which we could improve to 3.3. On the general graphs we could reduce the average ply number from 37.3 to 36.7 and on the planar subset we could even improve the average ply number from 10.4 to 8.8. The results are presented in Fig. 7.

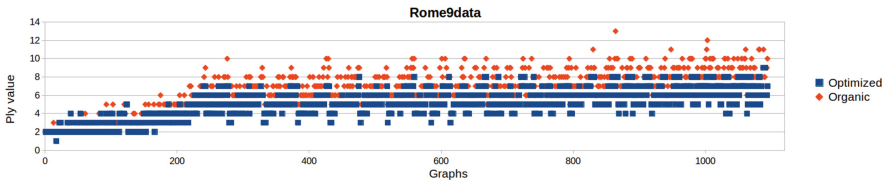


Fig. 6. For each graph in the **Rome9data** set the organic and the improved ply number are drawn. The graphs are ordered by the number of vertices.

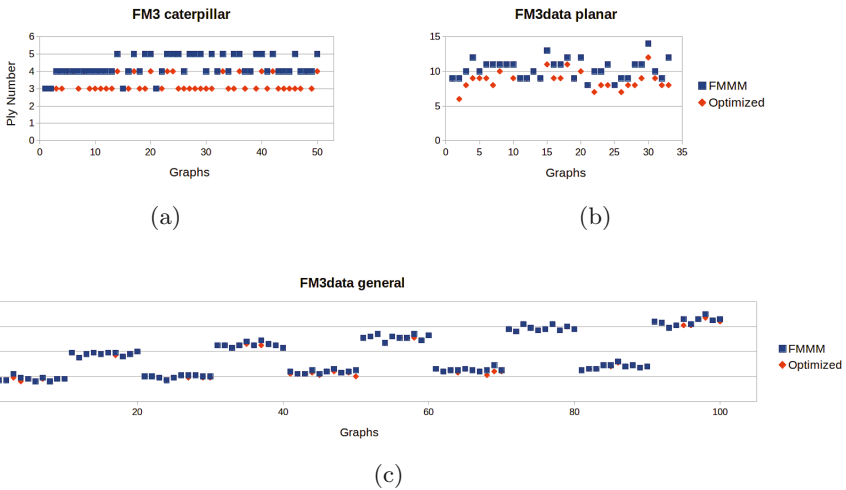


Fig. 7. The plots present the minimization results on **FM3data**. Note that the axis change in scale throughout the plots. (a) the ply number of the caterpillars. (b) the ply number of the planar graphs. (c) the ply number of the general graphs.

5 Discussion and Conclusion

To start our discussion we first want to analyze the results for the ply computation for the different layouts followed by the comparison between the two implementations. We continue with the ply minimization part. We conclude with a paragraph on the advantages of our tool.

We have presented the results of the ply computation on various graph layouts. Comparing different layouts, we can easily conclude that on sparse graphs spring embedding algorithms produce low ply drawings. This confirms the findings of [7]. Analyzing denser graphs, these algorithms tend to reach their limits. On very dense graphs (close to complete graphs) they perform similar to random layouts (see Fig. 3). To strengthen this claim we included the ply number for randomly drawn graphs in this figure. An interesting observation suggests that the circular layout produces ply numbers close to $\frac{|V|}{2}$ even in the worst case. The reason for this is stated in Sect. 4.3.

Our experiments suggest the equilibrium between the circular and the organic layout to be between density of 5 and 6.5. At densities larger than 6.5 the ply numbers for graphs drawn with the circular layout are clearly lower than the ply numbers for the organic layout (see Fig. 3).

The number of total events indicate a similar observation. While the number of events highly correlates with the increasing density, a higher number of events seems to imply a higher ply number of the drawing. Accordingly, the number of events in dense graphs support the observation that our organic layout produces similar ply numbers as the random layout (cf. Table 2). This effect is expected and the reason for this is twofold. On the one hand, every vertex has one ply disk which represents the dependency on the number of vertices. On the other hand, more edges tend to induce larger radii and thereby more intersections even though the number of disks stay the same. The increasing number of events according to the density and number of vertices is observable in both evaluated implementations.

According to the precision errors we observe a high number of **postponed events**, especially in the circular layout. This can be explained by the highly symmetric structure of these drawings, which cause many events to share an x-coordinate. The radii of the ply disks are likely to be irrational numbers and are thus prone for errors. Note that the value purely counts the number of events, which could not be solved instantly. There exist events which are counted several times, since they require more steps in between to be solved and we jump back to the first unsolved event.

Comparing the two implementations on the **FM3data**, we observe three facts. First of all the number of total events differ by a factor of 2 to 3. This can be explained due to the implementation of [7] adds additional events to prevent the influence of precision errors. This way the errors are detected instantly. Second, the difference in computation time depends partially on the pure number of events but the major time difference can be explained by the arithmetic computation time on `Apfloat` values in comparison to the primitive type `double`. Since we present a tool for the examination of different graphs we need

Table 5. The average results of the general graphs of **FM3data** are presented. The ply numbers in brackets indicate a different result of the algorithms. Note that these cases have a high number of postponed events.

Density	Vertices	Ply	[7]		Our Tool		
			Events	Time (ms)	Events	Time (ms)	postponed
1.5	250	18	18334	23955	6430	10.2	0.4
	300	19.8	25688.3	38454.8	8950.8	9.7	159
	350	23.7	34140.5	52829.1	11949.7	23.7	0.6
	400	25.4	43543.4	72928.5	15227.5	16.4	0.3
	450	28 (27.9)	55643.2	100653.2	19395.6	23.1	0.7
2.5	250	38.1	47248	92192.7	16539.1	19.5	0.5
	300	45.4	68070.5	147113.6	23892.3	36.7	2.2
	350	51.4	90943.4	217999.9	31850.1	43.5	2.5
	400	59.3 (58.7)	118188.7	309601.8	40606.9	83.8	40048.3
	450	64.3 (64.2)	148973.3	426993.5	51640.4	112.9	62776.7

a fast computation of the ply number to achieve a feedback for the user within milliseconds. Note that the implementation of [7] this was not applicable. To give an overview, there were only 4 out of 100 graphs where a difference in the computed ply number could be observed. In every case our implementation underestimated the ply in comparison to the other implementation [7]. The average error is very low as presented in Table 5.

Examining the results we can state a likelihood or quality of the computed result. Where the number of postponed events is one important indicator of occurred precision errors and evenly important a large number of end-events to solve inconsistencies increase the likelihood of miscomputation. A feature we want to include in future work is to give visual feedback to the user in that case.

During the experiments we detected a few computations with a high number of **postponed events** during the analysis of the **FM3data**. Examining the graphs, we observed that the FM3 algorithm tends to produce drawings with low average edge length and thereby are likely to induce precision errors. In our layout algorithms larger average edge length where possible. This increases the accuracy of our algorithm and explains the computation error on these graphs. All in all we present an algorithm which can compete in the computed result and is very fast. We conjecture that the accuracy can be even increased by scaling a given graph. Unfortunately, we cannot support this by experimental data, another task that will be tackled in future work.

Now we want to discuss the results of our minimization approach. We compare the organic layout and our strategy to reduce the ply number on **Rome9data**. We adjusted a spring embedder to reduce the ply number for a drawing based on our organic layout. One of the important observations on sparse graphs was that the maximal ply number is often reached in very few regions. On this data, in average, we can reduce the ply number by one. For fur-

ther competition on sparse graphs we compare the FM3 layout algorithm, which was the winning strategy in [7]. Our approach creates drawings that are on average one ply lower than this algorithm. (cf. Figure 7). Even though our implementation tends to underestimate the ply number on some **FM3data** graphs. Our modification produces a larger scaling and we conjecture that our approach constructs drawings with lower ply number and the computations are more resistant to precision errors.

For very dense graphs the spring embedding strategies seem to produce drawings which have a ply number close to random layouts. Nevertheless, for dense graphs we can guarantee an upper bound by using the circular layout, which is included in our optimization.

Examining the minimization strategies on caterpillars of **FM3data**, the ply numbers still range up to 4. Even though we know that caterpillars admit a ply 2 drawing [1]. Further examination on these graphs suggests that our methods are often able to construct drawings with ply number 2 given a suitable start configuration and enough time. Since we gave a strict time limit during the experiments we did not manage to produce many ply 2 drawings on this set.

Our tool provides the user with our adjusted spring embedder and the possibility to enforce equal edge lengths. The equal edge lengths can be interpreted as a test if the actual embedding admits a ply 1 drawing. During our experiments, due to precision errors, we did not observe ply 1 drawings by automated layout methods. The enforced equality of edge lengths includes very strong forces and converges if there exists a ply 1 drawing in the current embedding.

The optimization process involves several iterative computational steps using spring embedding algorithms and computation of the ply number in between. By using these steps and adjusting the vertices manually it is possible for the user to reduce the ply number even further by moving few vertices, since due to a previous observation there often exist only few regions with maximal ply.

We conclude this part with a short summary of functionality and a forecast for our tool. We introduced a fast ply computation algorithm which is able to give instant feedback to user interaction, e.g. whenever the drawing of a graph is modified. We were successfully able to reduce the computation time from seconds to milliseconds. Our tool is equipped with basic layout algorithms and simple automated minimization techniques. The tool can be used to get a deeper understanding of several graph classes e.g. according to the question if there exists a lower bound on the ply number. In the near future we will include an indicator on the accuracy of the computation. In these cases the implementation providing higher precision in the computation might be used as verification. Furthermore, we want to improve the minimization methods. Further evaluation and experiments will be necessary to observe the influence of scaling to our computations.

Acknowledgements. We specially thank the authors of [7] for providing their implementation and data to compare with ours. We also thank Patrizio Angelini, Lukas Bachus, Michael Bekos, and Felice De Luca for helpful discussions.

References

1. Angelini, P., Bekos, M.A., Bruckdorfer, T., Hančl, J., Kaufmann, M., Kobourov, S., Symvonis, A., Valtr, P.: Low ply drawings of trees. In: Hu, Y., Nöllenburg, M. (eds.) GD 2016. LNCS, vol. 9801, pp. 236–248. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_19
2. Angelini, P., Chaplick, S., De Luca, F., Fiala, J., Hancl, J., Heinsohn, N., Kaufmann, M., Kobourov, S., Kratochvíl, J., Valtr, P.: On vertex- and empty-ply proximity drawings. In: Proceedings of the 25th International Symposium on Graph Drawing and Network Visualization (GD 2017) (2017, to appear)
3. Atallah, M.J.: Algorithms and Theory of Computation Handbook. CRC Press, Boca Raton (1999)
4. Bachus, L.: Ply, University of Tübingen. Bachelor thesis (2016)
5. Brandes, U., Eiglsperger, M., Lerner, J., Pich, C.: Graph markup language (GraphML). In: Tamassia, R. (ed.) Handbook on Graph Drawing and Visualization, pp. 517–541. Chapman and Hall/CRC (2013)
6. Chimani, M., Gutwenger, C., Jünger, M., Klau, G.W., Klein, K., Mutzel, P.: The open graph drawing framework (OGDF). In: Tamassia, R. (ed.) Handbook on Graph Drawing and Visualization, pp. 543–569. Chapman and Hall/CRC (2013)
7. De Luca, F., Di Giacomo, E., Didimo, W., Kobourov, S., Liotta, G.: An experimental study on the ply number of straight-line drawings. In: Poon, S.-H., Rahman, M.S., Yen, H.-C. (eds.) WALCOM 2017. LNCS, vol. 10167, pp. 135–148. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-53925-6_11
8. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice-Hall (1999)
9. Di Giacomo, E., Didimo, W., Hong, S., Kaufmann, M., Kobourov, S.G., Liotta, G., Misue, K., Symvonis, A., Yen, H.: Low ply graph drawing. In: Bourbakis, N.G., Tsihrintzis, G.A., Virvou, M. (eds.) 6th International Conference on Information, Intelligence, Systems and Applications, IISA 2015, Corfu, Greece, 6–8 July 2015, pp. 1–6. IEEE (2015)
10. Eppstein, D., Goodrich, M.T.: Studying (non-planar) road networks through an algorithmic lens. In: Aref, W.G., Mokbel, M.F., Schneider, M. (eds.) 16th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2008, Proceedings, 5–7 November 2008, Irvine, California, USA, p. 16. ACM (2008)
11. Fruchterman, T.M.J., Reingold, E.M.: Graph drawing by force-directed placement. *Softw., Pract. Exper.* **21**(11), 1129–1164 (1991)
12. Hachul, S., Jünger, M.: Drawing large graphs with a potential-field-based multilevel algorithm. In: Pach, J. (ed.) GD 2004. LNCS, vol. 3383, pp. 285–295. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31843-9_29
13. Hachul, S., Jünger, M.: Large-graph layout algorithms at work: an experimental study. *J. Graph Algorithms Appl.* **11**(2), 345–369 (2007)
14. Himsolt, M.: GML: A Portable Graph File Format. Universität Passau (1997). <http://www.fmi.uni-passau.de/graphlet/gml/gml-tr.html>
15. Kobourov, S.G.: Force-directed drawing algorithms. In: Tamassia, R. (ed.) Handbook on Graph Drawing and Visualization, pp. 383–408. Chapman and Hall/CRC (2013)
16. Tamassia, R., Liotta, G.: Graph drawing. In: Goodman, J.E., O’Rourke, J. (eds.) Handbook of Discrete and Computational Geometry, 2nd edn., pp. 1163–1185. Chapman and Hall/CRC (2004)

17. Tommila, M.: A C++ high performance arbitrary precision arithmetic package (2003). <http://www.apfloat.org/apfloat/>
18. Welzl, E., Di Battista, G., Garg, A., Liotta, G., Tamassia, R., Tassinari, E., Vargiu, F.: An experimental comparison of four graph drawing algorithms. *Comput. Geom.* **7**, 303–325 (1997)
19. Wiese, R., Eiglsperger, M., Kaufmann, M.: *yFiles* - visualization and automatic layout of graphs. In: Jünger, M., Mutzel, P. (eds.) *Graph Drawing Software*, pp. 173–191. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-642-18638-7_8

Experimental Analysis of the Accessibility of Drawings with Few Segments

Philipp Kindermann¹(✉), Wouter Meulemans², and André Schulz¹

¹ LG Theoretische Informatik, FernUniversität in Hagen, Hagen, Germany
{philipp.kindermann, andre.schulz}@fernuni-hagen.de

² Department of Mathematics and Computer Science, TU Eindhoven,
Eindhoven, The Netherlands
w.meulemans@tue.nl

Abstract. The visual complexity of a graph drawing is defined as the number of geometric objects needed to represent all its edges. In particular, one object may represent multiple edges, e.g., one needs only one line segment to draw two collinear incident edges. We study the question if drawings with few segments have a better aesthetic appeal and help the user to assess the underlying graph. We design an experiment that investigates two different graph types (trees and sparse graphs), three different layout algorithms for trees, and two different layout algorithms for sparse graphs. We asked the users to give an aesthetic ranking on the layouts and to perform a furthest-pair or shortest-path task on the drawings.

1 Introduction

Algorithms for drawing graphs try to optimize (or give a guarantee) on certain formal quality measures. Typical measures include area, grid size, angular resolution, number of crossings, and number of bends. While each of these criteria is well motivated, we have no guarantee that we get a good drawing by optimizing only one of the measures. This is due to the fact that most measures compete with each other. For example, it is known that certain planar graphs cannot be drawn with good angular resolution and polynomial area [8]. The question arises how we can select an appropriate algorithm for a graph drawing task. Instead of relying on a combinatorial or geometric measure of the drawing, one could also value the results of the algorithms by measuring the efficiency of tasks carried out by the observer. Another option would be to just ask the observer which drawing he considers as “nice”. By conducting such experiments we also hope to learn something about the formal measures. The goal is here to identify formal measures/algorithms that are particularly suitable for typical tasks performed on graph drawings.

This work was partially funded by the German Research Foundation (grant SCHU 2458/4-1). W. Meulemans is funded by the Netherlands eScience Center (NLeSC, grant 027.015.G02).

In this paper, we present a study that investigates how good drawings with few segments are perceived by the observer in contrast to other drawing styles. A drawing with few segments tries to draw several edges that form a path as a single segment. Although the path may contain many edges, this is counted as only one segment in the drawing. The total number of segments is known as the *visual complexity* of the drawing. Instead of straight-line segments, one could also use other geometric objects to draw paths. One option that has been introduced by Schulz [12] is using circular arcs. However, in this paper our focus lies on drawings with segments.

It is an open question whether a small number of segments is a good quality measure for graph drawings. In our study, we want to find out if this design criterion makes drawings more aesthetically appealing for the observer and/or if they are helpful for executing tasks. The main difficulty is that we cannot control the visual complexity of a drawing while keeping other measures fixed. One way to avoid this problem is to adapt existing algorithms in such a way that we can reduce the number of segments in the final drawing without changing too much in the “*style*” of the existing drawing.

In our study, we focus on two graph classes. The first class are trees, for which many drawing algorithms are known. It is not hard to see that every tree can be drawn with $n_{\text{odd}}/2$ segments, where n_{odd} denotes the number of odd-degree nodes in the tree [3]. It is however unknown if every tree can be drawn with $n_{\text{odd}}/2$ segments using only a polynomial grid size. We heuristically improve the algorithm of Hültenschmidt et al. [6] that draws a tree with minimal visual complexity and quasi-polynomial area and compare its drawing in the user study against drawings of other algorithms. In particular, we use the algorithm of Walker II [14] and of Rusu et al. [11] as alternatives. The former one mimics the standard way how trees are typically drawn in the computer science literature. The latter one aims to draw trees with good angular resolution on a small grid.

The second class of graphs we consider are sparse but not necessarily plane graphs as provided by the *ROME library* [15]. In this setting, it is even harder to control more than one formal measure. We therefore selected only one algorithm to compare with. This is the popular Fruchterman-Reingold spring embedder algorithm [4]. In order to generate drawings that have a “*similar feel*” but use fewer segment, we adapt the spring embedder algorithm by adding constraints that force certain edges to be collinear, and hence form a straight-line segment.

We selected two categories for the users to evaluate the drawings presented to them. The first category addresses the question which of the drawings are aesthetically more appealing to the user. In the second category, we asked the users to perform tasks. For the trees, we asked to identify pair of nodes realizing the largest distance; for the sparse graphs we asked to select a shortest path between two designated vertices. The user study was implemented as a voluntary online questionnaire in order to reach a significant number of participants.

2 Algorithms

Trees. For trees, we used three algorithms as illustrated in Fig. 1: *Tidier*, *Quad*, and *FewSegments*. The algorithm *Tidier* was presented by Walker II [14] and

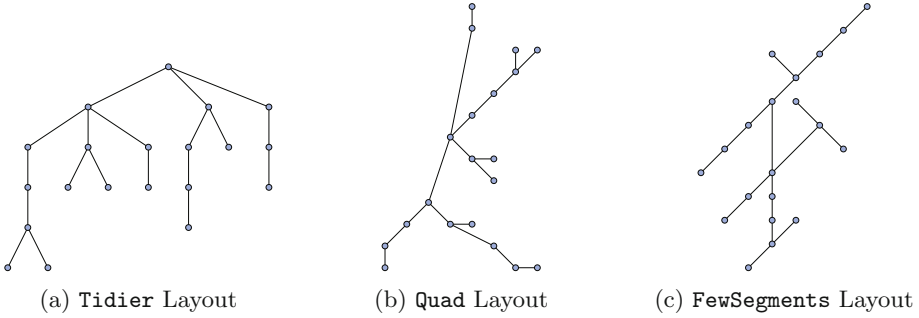


Fig. 1. A drawing of a tree with each of the three considered layouts.

builds upon the classic algorithm by Reingold and Tilford [10]. This algorithm satisfies three criteria: (1) Nodes at the same level of the tree should lie along a straight line, and the straight lines defining the levels should be parallel. (2) A parent should be centered over its offspring. (3) A subtree should be drawn the same way regardless of where it occurs in the tree.

The algorithm **Quad** was presented by Rusu et al. [11]. This algorithm allows the user to specify an angular coefficient and draws edges such that the angles are above the angular coefficient if possible and evenly spread out otherwise. It also allows the user to specify how many quadrants may be used to place the children of a vertex. We chose an angular coefficient of 22.5° and allowed the algorithm to use all four quadrants.

Finally, the algorithm **FewSegments** is based on the algorithm by Hülten-schmidt et al. [6] that draws trees on a quasi-polynomial grid with a minimum number of segments. On a high level, that algorithm uses a heavy path decomposition of a tree and places the heavy paths onto segments. It recursively embeds a subtree such that the heavy path of its root is drawn with a vector specified by the parent edge of its root and all subtrees lie in disjoint boxes. We use three heuristics to reduce the size of the drawing.

The first heuristic is applied during the layout of the tree. When the algorithm assigns a vector to a subtree, we allow it to increase the length of the vector slightly such that the new vector is an integer multiple of a smaller primitive vector. For example, if the algorithm would assign a vector $(6, 11)$, then this heuristic would change the vector to $(6, 12)$. This implies that the segments on the heavy path in this subtree do not have to use vectors that are integer multiples of $(6, 11)$, but only integer multiples of $(1, 2)$. Although this makes one segment a bit longer, the subtree might use less area by this change.

The second and the third heuristic are applied in alternating order after a layout has been found. The second heuristic tries to *compress* vectors: given a segment s that is drawn as a vector \vec{v} that is an integer multiple of a primitive vector \vec{u} , it redraws the tree such that s is drawn with the smallest integer multiple of \vec{u} without destroying planarity. The third heuristic takes a segment t that is drawn with a long vector \vec{w} and tries to find a smaller vector \vec{w}' to draw t

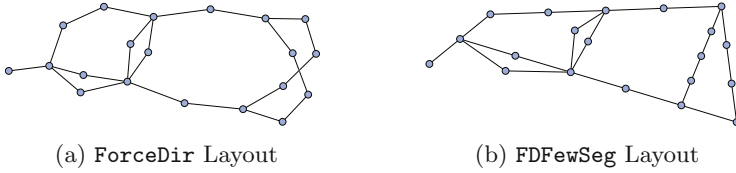


Fig. 2. A drawing of the ROME graph *2282.20* with both force-directed layouts.

and the subtree rooted in its child such that the resulting drawing is still planar. This is a more drastic approach and can change the way a subtree is drawn completely. We apply both post-processing heuristics five times on the resulting layout. The second heuristic is applied to all edges, the third only on those whose length is at least $1/5$ of the length of the diagonal of the minimum bounding box of the drawing.

Graphs. For the sparse graphs, we used the algorithms `ForceDir` and `FDFewSeg`; example drawings are provided in Fig. 2. The former is an implementation of the spring embedder by Fruchterman and Reingold [4]. This algorithm computes a force between each pair of vertices. If there is an edge between two vertices, then there is an *attractive force* $f_a(d) = d^2/k$ between them, where d is the distance between the vertices and k is their optimal distance defined as $k = C\sqrt{A/n}$, where C is some constant, A is the maximum area of the drawing, and n is the number of vertices in the graph. If there is no edge between two vertices, then there is a *repulsive force* $f_r(d) = -k^2/d$ between them. By an addition of these forces at every vertex v , we obtain a *movement* Δv of the vertex described by a 2-dimensional vector. Fruchterman and Reingold use simulated annealing to control the movement of the vertices such that the adjustments become smaller over time and the algorithm terminates.

The `FDFewSeg` algorithm is an extension of the `ForceDir` spring embedder. It takes as an additional input a set of edge-disjoint paths \mathcal{P} . First, the movement Δv for every vertex is computed. Let $v_0, \dots, v_k = P \in \mathcal{P}$. The algorithm places the vertices v_1, \dots, v_{k-1} evenly spaced onto the segment between v_0 and v_k . To this end, for every vertex $v_i, 0 < i < k$, the movement becomes

$$\Delta v_i = \frac{n-i}{n} (v_0 + \Delta v_0) + \frac{i}{n} (v_k + \Delta v_k) - v_i.$$

Note that this procedure does not necessarily draw all paths in \mathcal{P} as segments: if a vertex u of a path is an internal vertex of another path that is processed later, then u will be moved away from its path segment. Hence, the user should input that paths in an order that avoids this problem.

3 Hypotheses

We design a user study to compare aesthetics and legibility of drawings produced by the above-described algorithms. In particular, we pose and analyze for the following four hypotheses:

- H1.** For trees, the aesthetics ranking is **Tidier** > **FewSegments** > **Quad** for people with mathematics or computer science background and **FewSegments** > **Tidier** > **Quad** for people from a different background.
- H2.** For trees, path finding is easiest with the **FewSegments** layout, followed by **Tidier**, and hardest with **Quad**.
- H3.** For sparse graphs, the **ForceDir** layout is more aesthetically pleasing than the **FDFewSeg** layout.
- H4.** For sparse graphs, path finding is easier with the **FDFewSeg** layout than with the **ForceDir** layout.

Underlying the Hypotheses H2 and H4 is the idea that placing paths onto few segments makes it easier for the user to follow a path between two nodes since the eye only has to move along few directions and can traverse several nodes quickly along a segment. Evenly spacing out the nodes along a path in the force-directed layout should help the reader to quickly determine the number of nodes on a segment and thus to judge the combinatorial length of such a path.

For Hypothesis H1, we guess that the uniformity of the **Tidier** and the **FewSegments** layout would make them win over the **Quad** layout. For mathematicians or computer scientists, we expect that the **Tidier** layout wins since it creates a drawing in the standard way that trees are drawn in the literature. For people with different background, we expect that the **FewSegments** layout wins because it seems to be more schematic.

For Hypothesis H3, we think that the smooth curves in the **ForceDir** layout look nicer to a reader than the drawings of the **FDFewSeg** layouts because the latter ones can have sharp corners at the meeting point of two path segments; for example, Bar and Neta [1] argue that sharp corners have a negative effect on aesthetics as such bends are identified with threat. On the other hand, Vessel and Rubin [13] studied the objectiveness of taste—their conclusion is that there is typically agreement for natural images, abstract depictions are influenced more by individual taste. Though they cannot fully be eliminated, we believe that the uniformity of graphical presentation may mitigate personal preferences to allow for investigating an overall agreement in aesthetics.

4 Experimental Design

Selecting tasks. We used two tasks: **Aesthetics** and **Query**. We created different graphs for each task. For the Aesthetics task, we showed the user one drawing for each layout of the same graph next to each other. The order of the drawings was determined randomly. The user was asked to determine a ranking on the aesthetics of the drawings by clicking on them in the desired order.

We used different Query tasks based on the graph class. We showed the user one drawing at a time. Over time, every graph was presented the user once with each layout.

For the sparse graphs, we asked the users to find the shortest path between two randomly marked vertices that have distance at least 3 (the pair of vertices

was the same for each layout and each user). The user solved this task by clicking on the vertices (or edges) in the order that they appear on this path. To make sure that a user does not get stuck on a question, we allowed them to submit their answer even if no path was found. We helped the user with this task by marking (in a different color) the valid nodes and edges they can click on, which are those that are adjacent to the endpoint of one of the two paths starting in the two marked vertices.

For trees, shortest paths are uniquely defined which makes it unsuitable as a task. Hence, we asked the user to find the furthest pair of vertices, that is, the pair of vertices such that the distance between them is maximized. This also requires the user to inspect several paths in the graph. The user then had to click on the vertices that they determined as the furthest pair.

Generating stimuli. For trees, we have the following two variables for the stimuli:

- **Size.** Two different sizes: (1) 20 nodes and (2) 40 nodes.
- **Depth.** Three different tree depths as defined by the length of the longest root–leaf paths: (D) *deep* trees of depth 8 for size 1 and of depth 14 for size 2, (B) *balanced* trees of depth 5 for size 1 and of depth 9 for size 2, and (W) *wide* trees of depth 3 for size 1 and of depth 5 for size 2.

To construct random trees of given size and depth, we create a uniformly distributed random Prüfer sequence [9] and check whether the corresponding labeled tree has the given depth. It is known that Prüfer sequences provide a bijection between the set of labeled trees on n vertices and the set of sequences of $n - 2$ integers between 1 and n . Hence, this algorithm gives us uniformly distributed random trees of a given depth. For each size and depth, we created four different graphs for the Aesthetics task and two different graphs for the Query task. This gives us $2 \cdot 3 \cdot 4 = 24$ graphs for the Aesthetic tasks (4 repetitions) and $2 \cdot 3 \cdot 2 = 12$ graphs for the Query task (2 repetitions).

For the sparse graphs, we have the following two variables for the stimuli:

- **Size.** Two different sizes: (1) 30 nodes and (2) 60 nodes.
- **Type.** Two different types: (A) graphs from the ROME library and (B) random graphs.

For graphs of type A, we randomly picked graphs of the given size from the ROME library [15] that consists of 11,535 sparse, but not necessarily planar, graphs with 10 to 100 vertices. For graphs of type B, we created a random graph by creating a number of nodes specified by the size and picking 30 random edges for graphs of size 1 and 60 random edges for graphs of size 2; we used the resulting graph if and only if it is connected. For each size and type, we again created four different graphs for the Aesthetics task and two different graphs for the Query task. This gives us $2 \cdot 2 \cdot 4 = 16$ graphs for the Aesthetic tasks (4 repetitions) and $2 \cdot 2 \cdot 2 = 8$ graphs for the Query task (2 repetitions).

We created the graphs as JSON files that contained the coordinates of the vertices and the set of edges. During the study, the graphs were drawn using

the JavaScript library *D3.js* [2] as SVG figures to allow arbitrary resizing. The nodes were drawn using blue circles. Links were drawn in black with a small halo to increase separability between crossing links. The selected vertices and links in both Query tasks were marked in green and the selectable vertices and links in the shortest path task were marked in light blue.

Further considerations. For trees, we created 24 stimuli for the Aesthetics task and 36 stimuli for the Query task (one per graph and layout). For the sparse graphs, we created 16 stimuli for the Aesthetics task and 16 for the Query task. This gives us 92 stimuli in total. This is beyond what is reasonable for an online study, assuming 15 to 25 s per trial. Since the study has two different graph classes with different tasks, we used the graph class as a between-subjects measure. This still leaves 60 stimuli for the tree tasks. Since the size of a graph is very likely to be an overall factor by the larger difficulty of the Query task on a larger graph, we used the size as an additional between-subjects measure for trees. This way, we obtain three groups of stimuli: (1) 30 stimuli for trees of size 1, (2) 30 stimuli for trees of size 2, and (3) 32 stimuli for sparse graphs. A pilot study showed a completion time of about 15 min for each group.

We first show the Aesthetics task and then the Query task. We did this such that the user does not get a bias for a specific drawing style based on the difficulty of the Query task and instead of the most aesthetic one picks the one that they preferred in the Query section. Though explicitly asking for visual preference could bias performance in the following Query section, we expect this effect to be negligible as only one drawing is shown at any given time; and in any case less strong than the potential bias if the sections were to be inverted.

In order to account for learning effects, the order of the stimuli for each task was randomized for each participant. Before each stimulus, the participant was given a pause screen to reduce memory effects and at the same time allow them to pace themselves and reduce the possible impact of interruptions. The participants received one example question with an answer revealed after providing one, from which they could go back to task description, to ensure that the task was understood before starting the actual questions. We opted not to provide a longer series of training questions to keep time investment to a minimum.

Setup. We developed our user study with PHP and the JavaScript library *D3.js*. The study was hosted on a web server¹ and the data was stored in a MySQL database. Since the questionnaire was conducted online, we had no control over many parts of the experimental environment, e.g., device, pointing device, operating system, browser, screen size, interruptions. We asked the participants to fill in the questionnaire using a desktop or laptop computer, not a tablet or phone, and to use the pointing device they are most comfortable with. To make sure that the browser is suitable to run the questionnaire, the users first had to set a slider to the value depicted in an SVG figure. We could not control the screen size, resolution, or distance of the participant to their screen, so we let the user

¹ <http://tutte.fernuni-hagen.de/web/userstudy/fewarcs>.

control the scale of the web page by providing a *Shrink* and a *Grow* button. Further, we asked them to put their browser in full-screen mode to reduce distractions. We requested the participants to not engage in other activities during the questionnaire and to minimize interruptions, and to specify if any interruptions occurred at the end of the study.

We recruited the voluntary participants of the user study using a mix of mailing lists, social networks, and social media. Some background and preference information was asked upon completion, although this remained optional for what may be considered sensitive information (age, gender, country of residence).

5 Results

The data set for the analysis as well as all stimuli have been made available online². In total, 84 people volunteered and completed the online questionnaire, which was open for participation for two weeks. We inspected all comments left by participants. One participant had a longer break during one of the questions, rendering this particular question unsuitable for the analysis. As to maintain a balanced design to allow for stronger analysis methods, we excluded him from the analysis. This gave us 21 participants for both group 1 and group 2, and 41 participants for group 3. Of the 83 participants, 75 provided their age with an average of 36.96. In terms of country of residence, a majority of the participants live in Europe (63), predominantly in Germany (42).

Hypothesis H1. For the tree aesthetics task, we had 42 participants from groups 1 and 2 and each of them was shown 12 stimuli. This gave us a total of 504 rankings between the three layouts. We used loglinear Bradley-Terry (LLBT) modeling [5] of the 1,512 pairwise aesthetic preference comparisons to produce ranked *worth* scores for each of the three layouts. The worth score allows the consistency of preference to be assessed in forming an overall ranking of the three classes. Figure 3 shows the ranking of the three layouts in terms of aesthetic preference, broken down by the balance of the graph and by the background of the participants.

Consistently, the **Quad** layout was considered as the least aesthetic tree layout. Over all answers, the **Tidier** layout performed the best. There was some effect based on the balance of the graph. For each balance, we received 168 rankings. For *balanced* trees, the layouts **FewSegments** (worth score 0.4308) and **Tidier** (worth score 0.4302) were perceived equally aesthetic. However, for *deep* and *wide* trees the **Tidier** layout performed better than **FewSegments** with worth scores of 0.4757 versus 0.3785 (deep) and 0.4881 versus 0.3959 (wide).

The hypothesis was split into two parts, depending on the background of the participants. Let us first consider the participants with a mathematics or computer science background. There were 48 rankings by four people with a mathematics background, but not computer science; for those, the **Tidier** layout was

² <http://tutte.fernuni-hagen.de/web/userstudy/fewarcs/studyresults.html>.

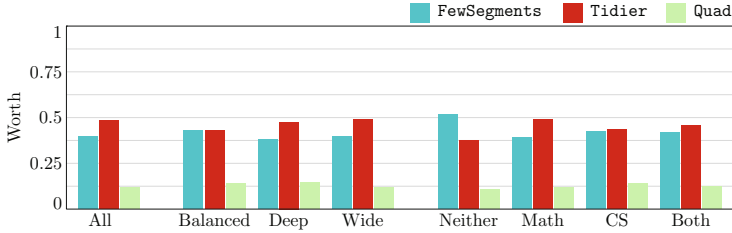


Fig. 3. Worth scores of the three tree layout methods: overall and partitioned by tree balance or participant background.

clearly preferred with a worth score of 0.4874 over 0.3929. There were 312 rankings by participants with a computer science background, but not mathematics; for those, the layouts were perceived similarly with a worth score of 0.4360 for *Tidier* and 0.4251 for *FewSegments*. There were 120 rankings by users with both mathematics and computer science background; those slightly preferred the *Tidier* layout with a worth score of 0.4559 over 0.4196. Overall, this suggests that there is some preference of the *Tidier* layout over the *FewSegment* layout, and of both layouts over the *Quad* layout; hence, we tentatively accept the first part of Hypothesis H1, due to the caveat of a small number of participants with a math-only background.

The second part of Hypothesis H1 is about participants from neither mathematics nor computer science background. There is some evidence for the hypothesis to be true. The participants from neither background strongly preferred the *FewSegments* layout over the *Tidier* layout with a worth score of 0.5173 over 0.3761. However, this only included 24 rankings from 2 participants, and thus the findings are a suggestion at best. This also indicates that our method for recruitment was not sufficiently broad enough to recruit volunteers from a variety of backgrounds. Indeed, this affects the other of the results as well.

Hypothesis H2. For the tree query task, we had 42 participants from groups 1 and 2 and each of them was shown 18 stimuli. This gave us a total of 756 tasks between the three layouts with 252 tasks per layout. We analyzed the error rates for finding a furthest pair for the three tree layouts defined by the difference of the distance between the picked pair and the distance between a furthest pair in the graph, broken down by the balance and by the size of the trees. The maximum response time was 53 s, so we did not have to exclude any participants. Figure 4 shows the error rates and the answer times by the participants.

We used a two-way RM-ANOVA to analyze the effects of the layouts, tree balance, tree size, and their interaction. We used the logarithm of the response times to normalize the distribution. For the error rate, there are no interaction effects between layout, balance, and size. The analysis showed a weak effect of the layout on the error rate ($F(2, 80) = 3.636, p < 0.05$). A post-hoc Tukey HSD test with Bonferroni adjustment showed a significant difference between layout *Quad* and *FewSegments* in favor of *FewSegments* ($p < 0.01$) and a significant

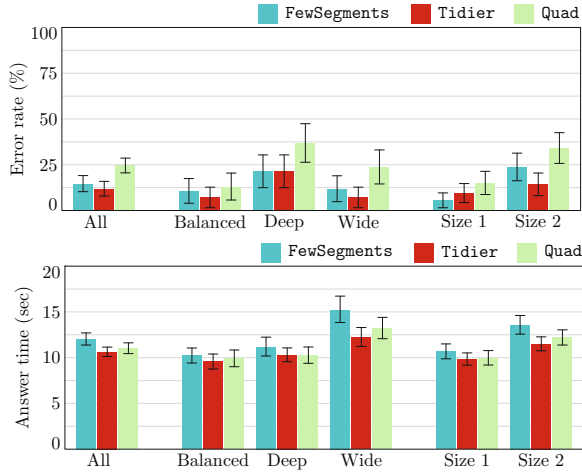


Fig. 4. Error rates and answer times for finding a furthest pair for the three tree layout methods: overall and partitioned by tree balance or size group. Error bars indicate 95% confidence intervals.

difference between layout **Quad** and **Tidier** in favor of **Tidier** ($p < 0.001$), but no significant difference between layout **Tidier** and **FewSegments**. Further, a post-hoc test showed a weak difference between the tree sizes ($p < 0.05$) in favor of smaller trees. For small trees, there is some evidence that **FewSegments** outperforms **Tidier** ($p < 0.05$); for large trees the error rate seems lower for **Tidier**, though no statistically significant effect was found ($p > 0.15$). We conclude that the layouts **FewSegments** and **Tidier** perform better than the layout **Quad**, while the participants performed better on small trees than on large trees.

For the response time, there is some interaction between tree size and tree balance ($F(4, 160) = 2.524$, $p < 0.05$), so we split according to sizegroup for further analysis. For small trees, there are no interaction effects between layout and tree balance. The analysis showed a very weak effect of layout ($F(2, 40) = 2.523$, $p < 0.1$) on response time. A post-hoc test showed a very weak difference between layout **Tidier** and **FewSegments** in favor of **Tidier** ($p < 0.1$) and no significant difference between the other two layout pairs. We conclude that the participants performed slightly faster for the **Tidier** layout than for the **FewSegments** layout.

For large trees, there are also no interaction effects between layout and tree balance. The analysis showed significant effect of layout ($F(2, 40) = 9.667$, $p < 0.001$) on response time. A post-hoc test showed a weak difference between layout **Quad** and **FewSegments** in favor of **Quad** ($p < 0.05$) and a significant difference between layout **Tidier** and **FewSegments** in favor of **Tidier** ($p < 0.001$). We conclude that the participants performed slightly faster for the **Quad** layout than for the **FewSegments** layout and significantly faster for the **Tidier** layout than for the **FewSegments** layout.

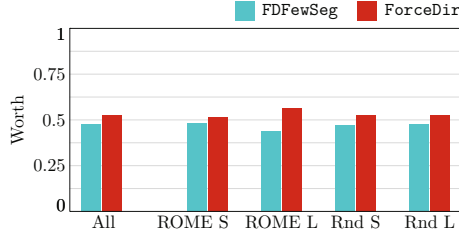


Fig. 5. Worth scores of the two layout methods: overall and partitioned by type.

Since the error rate was smaller for the `FewSegments` and `Tidier` layouts than for the `Quad` layout, but the response time for `FewSegments` was worse than for the other two, we can only partly accept Hypothesis H2: the layouts `FewSegments` and `Tidier` both outperform the layout `Quad`, but the layout `Tidier` outperforms the layout `FewSegments`. Though not initially hypothesized, we also found evidence of an effect of the tree balance on both the error rate and the response time (see the full version [7]) in favor of *balanced* and *wide*.

Hypothesis H3. For the sparse graph aesthetics task, we had 41 participants from group 3 and each of them was shown 16 stimuli. This gave us a total of 656 rankings between the two layouts. We again used LLBT modeling of the 656 pairwise aesthetic preference comparisons to produce ranked worth scores for both layouts. Figure 5 shows the ranking of the both layouts in terms of aesthetic preference, broken down by the graph class and the size of the graph.

Over all 656 rankings, the `ForceDir` layout was preferred with a worth score of 0.5246 over the `FDFewSeg` layout with a worth score of 0.4754. The `ForceDir` layout was preferred for each pair of graph class and size. The preference is the smallest for small graphs of the ROME library with a difference in worth score of 0.0316, and it is the largest for large graphs of the ROME library with a difference in worth score of 0.1267. Hence, we accept Hypothesis H3.

Hypothesis H4. For the sparse graph query task, we had 41 participants from group 3 and each of them was shown 16 stimuli. This gave us a total of 656 tasks between the two layouts with 328 tasks per layout. We analyzed the error rates for finding a shortest path for the two layouts defined by the difference between the length of the selected path and the length of a shortest path, broken down by the four graph types (ROME small, ROME large, Random small, Random large). Figure 6 shows the error rates and answer times by the participants.

We used the same analysis as for the tree query task. For the error rate, there is some interaction between layout and graph type ($F(3, 120) = 3.313$, $p < 0.05$), so we split according to graph type. For *Random small* graphs, we found a significant difference between the layouts in favor of `ForceDir` ($F(1, 40) = 9.949$, $p < 0.01$); for the other graph types, there is no significant effect of the layouts.

For the response time, there is a strong interaction between layout and graph type ($F(3, 120) = 21.06$), $p < 0.001$), so we split according to graph type.

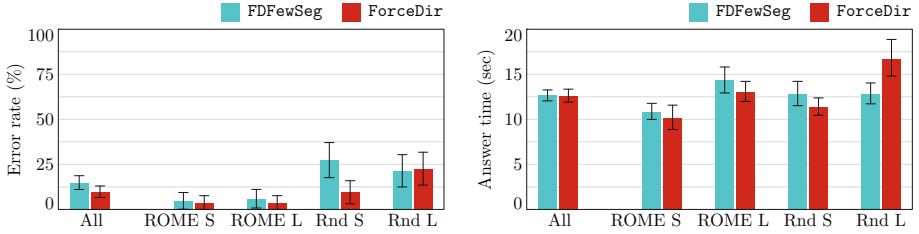


Fig. 6. Error rates and response times for finding a shortest path for the two layout methods: overall and partitioned by graph type. Error bars indicate 95% confidence intervals.

For *ROME small* ($F(1, 40) = 9.317$, $p < 0.01$) and *Random small* graphs ($F(1, 40) = 7.474$, $p < 0.01$), there is a significant effect of the layouts on the response time in favor of *ForceDir*. For *ROME large* graphs, there is a very weak effect of the layouts on the response time in favor of *ForceDir* ($F(1, 40) = 3.901$, $p < 0.1$). For *Random large* graphs, there is a significant effect of the layouts on the response time in favor of *FDFewSeg* ($F(1, 40) = 24.56$, $p < 0.001$).

Since the *ForceDir* layout outperformed the *FDFewSeg* on three of the four graph layouts, we have to reject Hypothesis H4 in general. However, the *FDFewSeg* performed better on large random graphs, so there is some evidence that this layout can give better results if the input graph has many vertices. The reason for this may be that the *ROME* graphs tend to have many degree-2 vertices. Similar effects can be observed for the small random graphs. Consequently, paths become easily traceable for these instances even if drawn with many segments.

6 Conclusion

We compared various graph layout algorithms to assess the effect of low visual complexity on aesthetics and performance. We have partially confirmed Hypothesis H1, that is, that for trees people with a math or computer science background tend to prefer the classical top-down layout. We also found some evidence that people with no such background prefer the layouts produced by the algorithm assuring low visual complexity; however, lacking a large number of participants in this category makes this only a suggestion of a possible effect. We have also partially confirmed Hypothesis H2, by finding evidence that finding a furthest pair is the easiest with the classical tree layout. We accepted Hypothesis H3 that for sparse graph the traditional force-directed layout is more aesthetic than its modification to reduce the visual complexity. We rejected Hypothesis H4 in general, but rather found that it is typically easier to find the shortest paths between two nodes with the traditional force-directed layout than the modification, though our hypothesis was found to hold for large random graphs. This leaves the possibility open that for graphs that are more intertwined using few segments can be beneficial.

In short, our findings suggest that visual complexity may positively influence aesthetics, depending on the background of the observer, as long as it does not

introduce unnecessarily sharp corners. Hence, drawings trees with few segments give a more schematic alternative over the classic drawing style without the risk of harming the aesthetic perception. However, few-segment drawings tend not to improve task performance. It is worth noting that we did not provide training to our participants, as to suggest how the segments can for example help to easily assess the length of a subpath. Providing such clues may have a positive effect on the performance, but at the same time would also result in an unfair comparison, if no training or strategies for the traditional layout were to be suggested.

Acknowledgments. The authors would like to thank all anonymous volunteers who participated in the presented user study.

References

1. Bar, M., Neta, M.: Humans prefer curved visual objects. *Psychol. Sci.* **17**(8), 645–648 (2006)
2. Bostock, M., Ogievetsky, V., Heer, J.: D³ data-driven documents. *IEEE Trans. Visual Comput. Graphics* **17**(12), 2301–2309 (2011)
3. Dujmović, V., Eppstein, D., Suderman, M., Wood, D.R.: Drawings of planar graphs with few slopes and segments. *Comput. Geom.* **38**(3), 194–212 (2007)
4. Fruchterman, T.M.J., Reingold, E.M.: Graph drawing by force-directed placement. *Softw. Pract. Exper.* **21**(11), 1129–1164 (1991)
5. Hatzinger, R., Dirrich, R.: prefmod: an R package for modeling preferences based on paired comparisons, rankings, or ratings. *J. Statist. Softw.* **48**(10), 1–31 (2011)
6. Hültenschmidt, G., Kindermann, P., Meulemans, W., Schulz, A.: Drawing planar graphs with few geometric primitives. In: Bodlaender, H.L., Woeginger, G.J. (eds.) *WG 2017. LNCS*, vol. 10520, pp. 316–329. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68705-6_24
7. Kindermann, P., Meulemans, W., Schulz, A.: Experimental analysis of the accessibility of drawings with few segments. Arxiv report [arXiv: 1708.09815](https://arxiv.org/abs/1708.09815) (2017)
8. Malitz, S.M., Papakostas, A.: On the angular resolution of planar graphs. *SIAM J. Discrete Math.* **7**(2), 172–183 (1994)
9. Prüfer, H.: Neuer Beweis eines Satzes über Permutationen. *Arch. Math. Phys.* **27**, 742–744 (1918)
10. Reingold, E.M., Tilford, J.S.: Tidier drawings of trees. *IEEE Trans. Software Eng.* **7**(2), 223–228 (1981)
11. Rusu, A., Yao, C., Crowell, A.: A planar straight-line grid drawing algorithm for high degree general trees with user-specified angular coefficient. In: *Proceedings of 12th International Conference Information Visualization (IV 2008)*, pp. 600–609. IEEE Computer Society (2008)
12. Schulz, A.: Drawing graphs with few arcs. *J. Graph Algorithms Appl.* **19**(1), 393–412 (2015)
13. Vessel, E., Rubin, N.: Beauty and the beholder: highly individual taste for abstract, but not real-world images. *J. Vis.* **10**(2), 1–14 (2010)
14. Walker II, J.Q.: A node-positioning algorithm for general trees. *Softw. Pract. Exper.* **20**(7), 685–705 (1990)
15. Welzl, E., Di Battista, G., Garg, A., Liotta, G., Tamassia, R., Tassinari, E., Vargiu, F.: An experimental comparison of four graph drawing algorithms. *Comput. Geom.* **7**, 303–325 (1997)

Obstacles and Visibility

Obstacle Numbers of Planar Graphs

John Gimbel¹, Patrice Ossona de Mendez^{2,3} , and Pavel Valtr⁴ 

¹ Department of Mathematics and Statistics, University of Alaska Fairbanks, Fairbanks, USA

jggimbel@alaska.edu

² Centre d'Analyse et de Mathématiques Sociales (CNRS, UMR 8557), Paris, France
pom@ehess.fr

³ Computer Science Institute of Charles University (IUUK), Prague, Czech Republic

⁴ Department of Mathematics of Charles University (KAM) and CE-ITI, Prague, Czech Republic
valtr@kam.mff.cuni.cz

Abstract. Given finitely many connected polygonal obstacles O_1, \dots, O_k in the plane and a set P of points in general position and not in any obstacle, the *visibility graph* of P with obstacles O_1, \dots, O_k is the (geometric) graph with vertex set P , where two vertices are adjacent if the straight line segment joining them intersects no obstacle.

The *obstacle number* of a graph G is the smallest integer k such that G is the visibility graph of a set of points with k obstacles. If G is planar, we define the *planar obstacle number* of G by further requiring that the visibility graph has no crossing edges (hence that it is a planar geometric drawing of G).

In this paper, we prove that the maximum planar obstacle number of a planar graph of order n is $n - 3$, the maximum being attained (in particular) by maximal bipartite planar graphs.

This displays a significant difference with the standard obstacle number, as we prove that the obstacle number of every bipartite planar graph (and more generally in the class PURE-2-DIR of intersection graphs of straight line segments in two directions) of order at least 3 is 1.

1 Introduction

Let O_1, \dots, O_k be closed connected polygonal obstacles in the plane, and let P be a finite set of points not in any obstacle. We assume that all the points in P and all vertices of the polygons O_1, \dots, O_k are in *general position*, that is that no three of them are on a line. Note that by considering some ϵ -neighborhood of the obstacles, we see that considering closed or open obstacles makes no difference (thanks to our general position assumption). Also, allowing or forbidding holes

P. Ossona de Mendez was supported by grant ERCCZ LL-1201 and by the European Associated Laboratory “Structures in Combinatorics” (LEA STRUCO).

P. Valtr was supported by project CE-ITI no. P202/12/G061 of the Czech Science Foundation (GAČR).

in obstacles makes no difference, as holes disappear if we drill a narrow passage from the outside boundary of an obstacle to each hole inside the obstacle.

The *visibility graph* of P with obstacles O_1, \dots, O_k is the geometric graph with vertex set P , where two vertices $u, v \in P$ are connected by an edge (geometrically represented by the segment uv) if the segment uv does not meet any of the obstacles. Visibility graphs have been extensively studied, see [5, 13, 17, 18, 23].

Closely related to visibility representation, Alpert *et al.* [1] introduced the *obstacle number* of a graph G , which is the minimum number of obstacles in a visibility representation of G . For instance, it is known that bipartite graphs can have arbitrarily large obstacle number [19]. Moreover it was shown in [6] that there are graphs on n vertices with obstacle number at least $\Omega(n/(\log \log n)^2)$. Although there is a trivial quadratic upper bound for the obstacle number, it is an open problem whether the obstacle number has a linear upper bound [1, 14]. An unexpected, almost linear upper bound ($2n \log n$) was recently obtained in [2]. For related complexity issues we refer the reader to [4, 20].

However the situation changes when one restricts his attention to planar graphs. Finding a planar graph with obstacle number greater than one was an open problem [1, 14] until it was recently proved [3] that the icosahedron has obstacle number 2. It is still an interesting open problem to decide whether the obstacle number of planar graphs can be bounded from above by a constant. This has been verified for outerplanar graphs by Alpert *et al.* [1], who proved that every outerplanar graph has obstacle number at most 1 (see also [12]). In this paper we complement this partial result by proving that planar bipartite graphs have obstacle number at most 1.

When considering a planar graph G , the existence of a planar visibility representation of G suggests another definition of the obstacle number. Here *planar visibility representation* means a geometric visibility graph in which no two edges cross (see Fig. 1). We define a new graph invariant, the *planar obstacle number* of G , as the minimum number of obstacles in a planar visibility representation of G . (Note that in this context, it will be convenient to assume that obstacles are *domains*, that are open connected sets.)

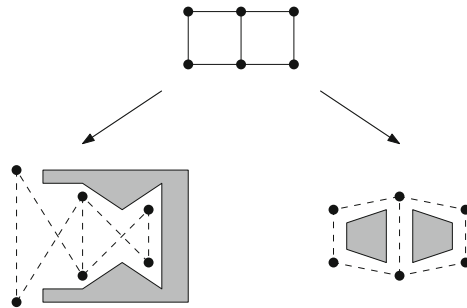


Fig. 1. Example of a graph having a visibility representation with one obstacle (on the left) and a planar visibility representation with two obstacles (on the right).

2 Our Results

For a planar graph G , the *planar obstacle number* of G , denoted by $\text{pobs}(G)$, is defined as the minimum number of obstacles needed to realize G by a straight-line planar drawing of G and a set of obstacles.

For $n \in \mathbb{N}$ we further define

$$\text{pobs}(n) = \max_{\substack{|G|=n \\ G \text{ planar}}} \text{pobs}(G).$$

The above definitions are correct due to Fáry's theorem [7] which says that every planar graph has a straight-line planar drawing; we call any such drawing a *Fáry drawing*. Slightly perturbing the set of vertices if necessary, we get a Fáry drawing with vertices lying in general position. Then it is possible to realize the graph by putting a small obstacle in each face, and this bound can be reduced by one if G is not a tree (in which case, for some embedding of G , one can remove the obstacle of the outer face). Hence for a graph G with n vertices and m edges this gives the bound $\text{pobs}(G) \leq \min(m - n + 1, 1)$.

Obviously, $\text{pobs}(2) = \text{pobs}(3) = 1$. In the following theorem we determine $\text{pobs}(n)$ for all $n \geq 4$.

Theorem 1. For $n \geq 4$,

$$\text{pobs}(n) = n - 3. \quad (1)$$

Also we can bound $\text{pobs}(G)$ by means of the number of edges, vertices and triangular faces. For a planar graph G , let $\text{tr-faces}(G)$ be the maximum number of triangular faces in a planar drawing of G . It is easily checked that one can drop an obstacle in any set of non-adjacent triangles, which improves the trivial upper bound to

$$\text{pobs}(G) \leq \min(m - n + 1 - \lfloor \text{tr-faces}(G)/3 \rfloor, 1).$$

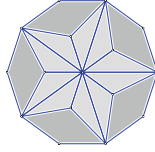
Theorem 2. For every connected planar graph G with $n \geq 5$ vertices and m edges, the following inequalities hold:

$$\max\{m - n + 1 - \text{tr-faces}(G), 1\} \leq \text{pobs}(G) \leq \max\{m - n + 1 - \lfloor \text{tr-faces}(G)/2 \rfloor, 1\}. \quad (2)$$

More precisely, if $\text{tr-faces}(G) \leq 1$ we have

$$\text{pobs}(G) = \begin{cases} \max\{m - n + 1, 1\} & \text{if } \text{tr-faces}(G) = 0 \\ \max\{m - n, 1\} & \text{if } \text{tr-faces}(G) = 1 \end{cases} \quad (3)$$

In the proof of this theorem, a key argument is that when two adjacent triangles have a non-convex union, the obstacles in these triangles may sometimes be dropped, but not always as witnessed by the following example:



Note that a linear time representation algorithm (starting from an embedding of G) is easily derived from our proofs. However, choosing an embedding in order to maximize $\text{tr-faces}(G)$ is difficult in general, as a reduction from minimum dominating set problem in planar graphs with maximum degree 3 shows that deciding (for input (G, k)) whether $\text{tr-faces}(G) \geq m - n - k$ is an NP-hard problem.

Planar obstacle number displays a significant difference with the standard obstacle number, as witnessed by the conjecture that planar graphs have bounded obstacle number.

Using a combinatorial characterization of visibility graphs with a single unbounded obstacle and vertices in convex position (Lemma 6) we determine the obstacle number of PURE-2-DIR graphs, where PURE-2-DIR graphs are intersection graphs of straight line segments with 2 directions such that any two segments belonging to a same direction are disjoint [15].

Theorem 3. *Let G be a PURE-2-DIR graph (i.e. a graph representable by intersection of straight line segments with 2 directions with no two segments in a same class intersecting). Then the obstacle number of G is either 0 (if G is K_1 or K_2) or 1 (otherwise).*

As it is known that every bipartite planar graph is a PURE-2-DIR graph, and more precisely has is the contact graph of a family of horizontal and vertical straight line segments in the plane [8] we immediately deduce from Theorem 3 the value of the planar obstacle number of bipartite planar graphs:

Theorem 4. *The obstacle number of a bipartite planar graph G is either 0 (if G is K_1 or K_2) or 1 (otherwise).*

3 Preliminaries

In this paper, we consider graphs that are finite, simple, and loopless.

A *drawing* of a graph G in the plane consists in assigning distinct points of the plane to the vertices of G , and arcs connecting points to edges. A drawing is *simple* if any two arcs intersect at most once, and it is *geometrical* if the arcs are straight line segments.

A *planar* graph is a graph that has a drawing in the plane in which no two arcs cross. Such a drawing is called an *embedding* of the graph. A planar graph embedded in the plane is called a *plane graph*. A *face* of a plane graph is a connected component of the complement of the drawing in the plane. A *face* is *bounded* (or an *inner face*) if it has a bounded diameter, it is *unbounded* otherwise. Note that every plane graph has exactly one unbounded face, which is called the *outer face*. The traversal of the boundary of a face then produces

a sequence of edges (precisely of arcs representing edges), whose length is the *length* of the face. Note that if the graph is 2-connected, the boundary of each face is a cycle, whose length is the length of the face (in general, bridges are counted twice). Each face of a (loopless simple) 2-connected graph has thus length at least 3. A *topological embedding* of a planar graph G in the plane is the equivalence class of an embedding of G in the plane under homeomorphisms of the plane. As they are invariant by every homomorphism of the plane, one can speak about the faces, inner faces, and outer face of a topological embedding of a planar graph in the plane. It is well known that a topological embedding in the plane is fully determined by the cyclic order of the incident edges around each vertex of the graph and the specification of the outer face. For more informations and background on topological graph embeddings we refer the reader to [16].

Every (simple) planar graph has a *Fáry drawing*, that is a crossing free geometric drawing [7] (see also [21]), and that such a drawing — with points in a linear size grid — can be computed in linear time [10, 11]. A planar drawing of a graph in which every face is a 3-cycle (i.e. has length 3) is a *triangulation*. A *maximal planar graph* is a planar graph G to which no edge can be added while preserving the planarity. It is easily checked that a planar graph is maximal if and only if some/every planar drawing of it is a triangulation.

By definition, every planar visibility representation of a graph G defines a Fáry drawing of G , each obstacle lying inside a single face of the drawing. It follows that any obstacle can be extended to any connected subset of the interior of the face. Thus, when minimizing the number of obstacles we may assume that every face f contains at most one obstacle, and that if f contains an obstacle, this obstacle is large enough to intersect every segment intersecting f which connects two independent vertices of G . Note that such a planar visibility representation is thus characterized by a Fáry drawing of G and the set of its faces containing an obstacle.

Observe that if $F(G)$ is a Fáry drawing of a connected planar graph G and H is an inner face of $F(G)$ which is not bounded by a 3-cycle then some straight-line diagonal of H lies entirely in H and therefore any realization of G using $F(G)$ must contain an obstacle lying inside the face H .

It follows from Tutte's spring theorem [22] that if a cycle C of a planar graph G bounds some face of some planar drawing of G , then G has a Fáry drawing such that the boundary of the outer face is the cycle C drawn as the boundary of a convex polygon.

4 Proof of Theorems 1 and 2

In this section we derive Theorems 1 and 2 from Lemma 3 stated below. Lemma 3 is then proved in the next section.

We first prove a weaker statement than Theorem 2, which determines the planar obstacle number for connected planar graphs not admitting more than one triangular face in any planar drawing, as given by (3), as well as the lower bound of (2).

Lemma 1. *For every connected planar graph G with $n \geq 5$ vertices and m edges, it holds*

$$\max\{m - n + 1 - \text{tr-faces}(G), 1\} \leq \text{pobs}(G) \leq \max\{m - n + 1, 1\},$$

where equality with the upper bound holds if and only if $m \leq n$ or $\text{tr-faces}(G) = 0$.

Thus if $\text{tr-faces}(G) \leq 1$ then

$$\text{pobs}(G) = \max\{m - n + 1 - \text{tr-faces}(G), 1\}.$$

Proof. For the lower bound, remark that every planar visibility representation defines a Fáry drawing with $m - n + 1$ inner faces. In this drawing, every inner face with length at least 4 has to contain an obstacle. Thus if $n \geq 5$ (so that $\text{pobs}(G) > 0$) it holds $\text{pobs}(G) \geq \max\{m - n + 1 - \text{tr-faces}(G), 1\}$.

If a planar graph G is acyclic, it obviously holds $\text{pobs}(G) \leq 1$. Otherwise, there exists a planar embedding of G with a face C which is a cycle. Using Tutte's spring embedding, there exists a Fáry drawing of G whose outer face is a convex polygon. Putting an obstacle in each of the $m - n + 1$ inner faces we get a planar visibility representation of G . Thus for every planar graph G on $n \geq 5$ vertices it holds

$$\text{pobs}(G) \leq \max\{m - n + 1, 1\},$$

and equality holds if and only if G has no embedding with a triangular face or $m \leq n$. Indeed, assume $m > n$. Consider an embedding of G with an inner triangular face $t = (v_1, v_2, v_3)$. At most one more face can contain all of v_1, v_2, v_3 , for if three faces would contain all of v_1, v_2, v_3 , adding a vertex in each of these faces adjacent all of v_1, v_2, v_3 would lead to a planar embedding of the non-planar graph $K_{3,3}$. As $m > n$ the embedding has at least three faces, including one face f which does not contain all of v_1, v_2, v_3 . We may assume that f is the outer face. As G is not acyclic, f includes (at least) one cycle C . For each cut-vertex u of C , we flip the connected components of $G - u$ that do not contain C into an inner face different from t (such a face exists, as otherwise f would contain all the three vertices of t). After this is done, the outer face is C and t is an inner face. Using Tutte's spring embedding one obtains a Fáry drawing of G , in which C is the outer face (drawn as a convex polygon) and t is an inner triangular face. Putting an obstacle in every inner face different from t we get a planar visibility representation of G with $m - n$ obstacles. \square

4.1 Three Lemmas

Here we state three lemmas used in the proof of both Theorems 1 and 2. We first give several definitions.

An edge e of a Fáry drawing of a planar graph G is said to be *concave* if it lies in two triangular faces and the union of these two faces is a non-convex quadrilateral. (If G is a triangulation then such an edge is sometimes called a *non-flippable edge* in the literature.)

Suppose T is a triangulation, x, y, z are three vertices of T , and X is a subset of the edge set of T . Then we say that a Fáry drawing of T is $(X; x, y, z)$ -concave if x, y, z are the vertices of the outer face and all the edges of X are concave.

For a graph $G = (V, E)$, an edge set $E' \subseteq E$ is said to be *sparse in G* if each cycle in G contains at least two edges of $E \setminus E'$.

Lemma 2. *For every triangulation T there is a 2-coloring of the faces of T such that the set of edges contained in a pair of faces of the same color is sparse in T .*

Proof. For a 2-coloring χ of the faces of T , let $S(\chi)$ be the set of edges contained in a pair of faces of the same color. Let $\bar{\chi}$ be a 2-coloring of the faces of T minimizing the size of $S(\bar{\chi})$, and let $F(T)$ be any fixed (topological) planar drawing of T . Any cycle C of T contains at most $\lfloor |E(C)|/2 \rfloor \leq |E(C)| - 2$ edges of $S(\bar{\chi})$, since otherwise the size of $S(\bar{\chi})$ could be decreased by flipping the colors of all the faces lying inside the cycle C in the drawing $F(T)$. Thus, $S(\bar{\chi})$ is sparse and $\bar{\chi}$ satisfies Lemma 2. □

Lemma 3. *Let T be a triangulation having a face with vertices a, b, c , and let $S \subseteq E(T)$ be a sparse set of edges of T . Then T has an $(S; a, b, c)$ -concave drawing.* □

The proof of Lemma 3 is sketched in Sect. 4.3; all details will be included in the full version. Importance of $(S; a, b, c)$ -concave drawings clearly appears in the following technical lemma.

Lemma 4. *Let T be a planar triangulation, let G be a spanning subgraph of T which includes all the edges of a face bounded by a triangle $\{a, b, c\}$ of T , let χ be a 2-coloring of the faces of T such that the set S of edges of T contained in a pair of faces of the same color is sparse, let $P(T)$ be an $(S; a, b, c)$ -concave drawing of T , and let $P(G)$ be the Fáry drawing of G obtained from $P(T)$ by the removal of all the edges of $E(T) \setminus E(G)$.*

Then by placing an obstacle in each face of $P(G)$ that is not a triangular inner face with color 2 we get a planar visibility representation of G .

Proof. Let x, y be a pair of non-adjacent vertices of G , and let F_1, \dots, F_t be the sequence of the faces of $P(G)$ intersected in this order by the segment xy (faces may appear more than once in this sequence), and let $\bar{F}_1, \dots, \bar{F}_t$ be their closure. We have to show that at least one of the faces F_1, \dots, F_t is not a triangular face colored 2. Suppose for contradiction this is not the case and that F_1, \dots, F_t are triangular faces colored 2.

- If $t = 1$ then F_1 is not a triangular face, contradicting the assumption.
- if $t = 2$, since $\bar{F}_1 \cap \bar{F}_2$ lies in S it is concave in $P(T)$ (and also in $P(G)$), and $\text{conv}(\bar{F}_2 \cup \{x\}) = \text{conv}(\bar{F}_1 \cup \bar{F}_2)$ is a triangle. This contradicts the assumption that the segment xy is included in $\bar{F}_1 \cup \bar{F}_2$, which necessarily implies that $\bar{F}_1 \cup \bar{F}_2$ is a convex quadrilateral.
- If $t \geq 3$ then all the edges $\bar{F}_1 \cap \bar{F}_2, \bar{F}_2 \cup \bar{F}_3$ lie in S and therefore the boundary of F_2 contains two edges in S , contradicting the assumption that S is sparse. □

4.2 The Proof

Here we derive Theorems 1 and 2 from Lemmas 2, 3 and 4. Note that according to Lemma 1 we need to prove only the upper bound of (2) to establish Theorem 2.

Let $n \geq 4$. By Euler's formula every Fáry drawing of $K_{2,n-2}$ has $n - 3$ inner faces. Each of them is a quadrilateral and therefore has to contain an obstacle. Thus, $\text{pobs}(n) \geq \text{pobs}(K_{2,n-2}) \geq n - 3$. In order to prove Theorems 1 and 2 it remains to prove that for a graph G on n vertices and m edges the following inequality holds: $\text{pobs}(n) \leq \min(n - 3, m - n + 1 - \lfloor \text{tr-faces}(G)/2 \rfloor)$.

First we show that it suffices to prove the upper bound for all connected planar graphs on $n \geq 4$ vertices. This will easily follow from the next lemma which relies on an invariant closely related to $\text{pobs}(G)$. The invariant $\text{pobs}'(G)$ is defined as the minimum number of obstacles lying in inner faces of a realization of G , where the minimum is taken over all possible realizations of G . It follows from the definition that $\text{pobs}'(G) = \text{pobs}(G)$ if every realization of G with $\text{pobs}(G)$ obstacles uses only obstacles in the inner faces. Otherwise we have $\text{pobs}'(G) = \text{pobs}(G) - 1$.

Lemma 5. *Let G be a disconnected planar graph, and let A_1, \dots, A_α be the components of G .*

$$\text{pobs}(G) = \begin{cases} \sum_{i=1}^{\alpha} \text{pobs}'(A_i) & \text{if } \exists A_j \text{ with } \text{pobs}(A_j) = \text{pobs}'(A_j) > 0, \\ 1 + \sum_{i=1}^{\alpha} \text{pobs}'(A_i) & \text{otherwise.} \end{cases}$$

Proof. For a realization of G with the given number of obstacles, suppose first that $\text{pobs}(A_j) = \text{pobs}'(A_j) > 0$ for some A_j . We realize the component A_j with $\text{pobs}(A_j) = \text{pobs}'(A_j)$ obstacles. Inside one of its obstacles we make small individual holes for the remaining components. Each component A_i , $i \neq j$, is realized in "its" hole using $\text{pobs}'(A_i)$ obstacles placed inside its inner faces. The boundary of the hole surrounds the drawing of A_i so that the surrounding obstacle functions for A_i as an obstacle in the outside face of A_i . If there is no component with $\text{pobs}(A_j) = \text{pobs}'(A_j) > 0$ then we proceed similarly as above, realizing the components inside individual holes of one big obstacle.

It remains to show that G cannot be realized with a smaller number of obstacles. Consider a realization of G , and let A_i be one of its components. If we consider only the edges of A_i , one or more obstacles must appear in at least $\text{pobs}'(A_i)$ inner faces of the drawing of A_i . Let F be one of these faces. The interior of F may contain other components of G but (at least) one of the obstacles inside F does not lie in any face of a component of G lying inside F , as otherwise a vertex of F would see a vertex of a component lying inside F . Assigning this obstacle to A_i , we get at least $\text{pobs}'(A_i)$ obstacles assigned to A_i and to no other component of G . This gives the lower bound $\text{pobs}(G) \geq \sum_{i=1}^{\alpha} \text{pobs}'(A_i)$. Suppose now that there is no A_j with $\text{pobs}(A_j) = \text{pobs}'(A_j) > 0$. Then there must be a component A_i incident to the outer face of the drawing of G . We have $\text{pobs}(A_i) = \text{pobs}'(A_i) = 0$ or $\text{pobs}(A_i) > \text{pobs}'(A_i)$. It follows that either we need an additional obstacle in the outer face of A_i which does not

lie inside another component of G , or at least $\text{pobs}(A_i) \geq \text{pobs}'(A_i) + 1$ faces of A_i contain obstacles and we may assign at least $\text{pobs}'(A_i) + 1$ obstacles to the component A_i . \square

Observe that $\text{pobs}'(C_4) = 1$ and $\text{pobs}'(G) = 0$ for any other connected graph G on at most four vertices. Therefore, due to Lemma 5 and to the trivial inequality $\text{pobs}'(G) \leq \text{pobs}(G)$, it suffices to prove the upper bounds in Theorems 1 and 2 for connected planar graphs (on $n \geq 4$ and $n \geq 5$ vertices, respectively).

Let G be a connected planar graph on $n \geq 4$ vertices. We want to show that $\text{pobs}(G) \leq n - 3$. Let $F(G)$ be a Fáry drawing of G . We distinguish the following two cases: either $F(G)$ has a triangular face, or no face of $F(G)$ is triangular.

In the latter case, $m \leq 2n - 4$ thus $m - n + 1 \leq n - 3$. Hence by Lemma 1 $\text{pobs}(G) \leq n - 3$.

In the rest of the proof we suppose that $F(G)$ has at least a triangular face $\{a, b, c\}$. By adding straight-line edges to $F(G)$ we obtain a Fáry drawing of a triangulation further denoted T .

Let χ be a 2-coloring of the faces of T satisfying Lemma 2, and let S be the set of edges of T contained in a pair of faces of the same color. Due to Lemma 3, T has an $(S; a, b, c)$ -concave drawing $P(T)$. Let $P(G)$ be the Fáry drawing of G obtained from $P(T)$ by the removal of all the edges of $E(T) \setminus E(G)$.

We denote the colors of χ by 1 and 2. We partition the set of the inner faces of $P(G)$ into the following three subsets:

- $I_1 :=$ the set of the triangular inner faces of $P(G)$ having color 1,
- $I_2 :=$ the set of the triangular inner faces of $P(G)$ having color 2,
- $I_3 :=$ the set of the non-triangular inner faces of $P(G)$.

Since $P(T)$ has $2n - 5$ inner faces and each face in I_3 is the union of at least two faces of $P(T)$, we have $|I_1| + |I_2| + 2|I_3| \leq 2n - 5$. It follows that $\min\{|I_1| + |I_3|, |I_2| + |I_3|\} \leq \lfloor (2n - 5)/2 \rfloor = n - 3$. Also we have $|I_1| + |I_2| = \text{tr-faces}(G) - 1$ and $|I_3| = m - n + 2 - \text{tr-faces}(G)$. Hence $|I_1| + |I_2| + 2|I_3| = 2(m - n + 1) - (\text{tr-faces}(G) - 1)$, and

$$\min\{|I_1| + |I_3|, |I_2| + |I_3|\} \leq m - n + 1 - \lfloor \text{tr-faces}(G)/2 \rfloor.$$

Without loss of generality we suppose that

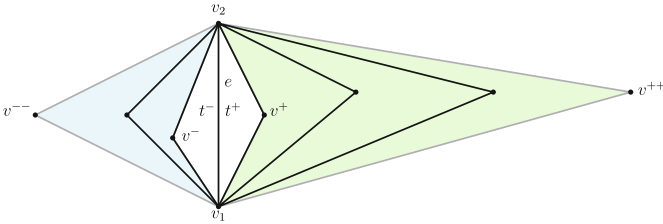
$$|I_1| + |I_3| \leq \min(n - 3, m - n + 1 - \lfloor \text{tr-faces}(G)/2 \rfloor).$$

To prove $\text{pobs}(G) \leq \min(n - 3, m - n + 1 - \lfloor \text{tr-faces}(G)/2 \rfloor)$ — from which Theorems 1 and 2 follow — it now suffices to prove that taking the Fáry drawing $P(G)$ and placing an obstacle in each face of $I_1 \cup I_3$ gives a representation of G , what follows from Lemma 4. This finishes the proof of Theorem 1. The proof of Lemma 3, which will be included in the full version of the manuscript, is only sketched here.

4.3 Sketch of the Proof of Lemma 3

We proceed by induction on n . The case $n = 3$ is trivial and the case $n = 4$ is easy. Suppose now that $n > 4$. If S is empty or contains only an edge of the

triangle abc then any Fáry drawing of T with the outer face abc is $(S; a, b, c)$ -concave. Suppose now that an edge $e \in S$ is not an edge of the triangle abc . Let Tr be the set of all triangles of T containing the edge e . We fix a Fáry drawing $F(T)$ of T with the outer face abc . The set Tr can be partitioned into two sets Tr^+ and Tr^- , such that Tr^+ contains the triangles of Tr lying on one side of e and Tr^- contains the triangles of Tr lying on the other side of e . Since T is a triangulation and e is an inner edge in $F(T)$, the set Tr^+ contains a unique face t^+ of $F(T)$. Similarly, Tr^- contains a unique face t^- of $F(T)$. Let t^{++} be the unique triangle in Tr^+ containing all the other triangles of Tr^+ in the drawing $F(T)$. The triangle t^{--} is defined analogously. Note that if $|\text{Tr}^+| = 1$ then $t^{++} = t^+$. Analogously, if $|\text{Tr}^-| = 1$ then $t^{--} = t^-$.



Without loss of generality, we may assume that in clockwise order one finds v^{++}, v_1, v_2 for the triangle t^{++} , v^+, v_1, v_2 for the triangle t^+ , v^-, v_2, v_1 for the triangle t^- , and v^-, v_2, v_1 for the triangle t^- .

We define a triangulation $T \odot e$ as the triangulation obtained from T by the planar contraction of e with respect to $F(T)$, i.e., it is the graph obtained from T by the following two operations: (i) removal of all the vertices and edges lying inside the triangles t^{++} and t^{--} in $F(T)$, and (ii) contraction of e . Thus, $e = v_1 v_2$ is contracted to a new vertex v_e and the triangles t^{++} and t^{--} (including their interiors in $F(T)$) are replaced by the new edges $v^{++} v_e$ and $v^{--} v_e$, respectively. The drawing $F(T)$ and the construction of $T \odot e$ immediately give a (topological) planar drawing $Dr(T \odot e)$ of $T \odot e$ with each face being a triangle and with the outer face abc . Let T^+ denote the triangulation consisting of the vertices and edges lying in the triangle t^{++} in the drawing $F(T)$. Analogously, let T^- denote the triangulation consisting of the vertices and edges lying in the triangle t^{--} in the drawing $F(T)$. Further, let $S^+ := S \cap E(T^+)$ and $S^- := S \cap E(T^-)$.

We can prove the following two observations: (1) S^+ is sparse in T^+ , and S^- is sparse in T^- ; (2) S_0 is sparse in $T \odot e$.

Hence we can apply the inductive hypothesis (i) on $T \odot e$ and S_0 , (ii) on T^+ and S^+ , and (iii) on T^- and S^- . In the first case we get an $(S_0; a, b, c)$ -concave drawing $P(T \odot e)$ of $T \odot e$ with the outer face abc . In the second case we get an $(S^+; v^{++}, v_1, v_2)$ -concave drawing $P(T^+)$ of T^+ with the outer face $t^{++} = v^{++} v_1 v_2$. In the third case we get an $(S^-; v^{--}, v_2, v_1)$ -concave drawing $P(T^-)$ of T^- with the outer face $t^{--} = v^{--} v_2 v_1$. Our construction of an $(S; a, b, c)$ -concave drawing of T is obtained by properly combining the drawings $P(T \odot e)$, $P(T^+)$ and $P(T^-)$. Starting with $P(T \odot e)$, we replace the vertex v_e by a short

segment e and the two edges $v_e v^{++}$ and $v_e v^{--}$ by skinny copies of $P(T^+)$ and $P(T^-)$, respectively, in such a way that the edge e is concave in the resulting drawing, thus proving Lemma 3 (details omitted here). \square

5 Obstacle Number of Intersection Graphs of Segments

Generally, it is an interesting question to characterize graphs with obstacle number 1, and specifically those graphs that can be represented using a single unbounded obstacle.

It is easily seen that a graph has such a representation if and only if it is an induced subgraph of the visibility graph of a simple polygon. However, no characterization is known for visibility graphs of simple polygons. However one can give some interesting characterization in a special case:

Lemma 6. *A graph G has a representation as a visibility graph with a single unbounded obstacle and vertices in convex position if and only if there exist functions p, I mapping the vertex set of G to points (resp. to circular arcs) of \mathbb{S}^1 , in such a way that*

- for every vertex v of G it holds $p(v) \notin I(v)$;
- for every distinct vertices u, v of G , it holds that u and v are adjacent if and only if $p(u) \in I(v)$ and $p(v) \in I(u)$.

Proof. The existence of a visibility representation from functions p and I is clear, and follows the same ideas as the representation for intersection graphs of horizontal and vertical segments.

Conversely, assume that G is representable as a visibility graph using a single unbounded obstacle, and consider such a representation. Without loss of generality, we can assume that the obstacle is the exterior of a simple polygon, on which lie the vertices of G . Let v_1, \dots, v_n be the vertices of G in the order in which they appear on the polygon (say clockwise), and let $i \neq j$. Let us now prove that v_i and v_j are adjacent if and only if there exist i_1, i_2 and j_1, j_2 such that

- in circular order, i is between i_1 and i_2 and j is between j_1 and j_2 ;
- v_i is adjacent to both v_{j_1} and v_{j_2} and v_j is adjacent to both v_{i_1} and v_{i_2} .

If v_i and v_j are adjacent, one can let $i_1 = i_2 = i$ and $j_1 = j_2 = j$. Conversely, if i_1, i_2, j_1, j_2 have the properties above, then let x (resp. y) be the intersection of segments $[v_i, v_{j_2}]$ and $[v_j, v_{i_1}]$ (resp. of segments $[v_i, v_{j_1}]$ and $[v_j, v_{i_2}]$). Then the region R delimited by the quadrangle (x, v_i, y, v_j) does not meet the obstacle, thus v_j is visible from v_i , that is v_i and v_j are adjacent (Fig. 2).

Let us now define functions p and I : for $1 \leq k \leq n$ let $p(v_k) = e^{ik2\pi/n}$ and $I(v_k)$ be the inclusion minimum circular arc containing all the neighbours of v_k but not v_k . It follows from the property above that the functions p and I satisfy the requirements of the Lemma. \square

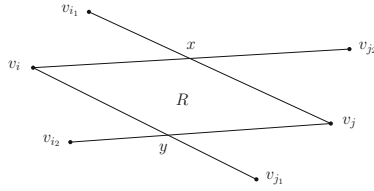


Fig. 2. Illustration for the proof of Lemma 6.

We now prove Theorem 3, which states that the obstacle number $\text{obs}(G)$ of a PURE-2-DIR graph G can be computed as follows:

$$\text{obs}(G) = \begin{cases} 0 & \text{if } G \text{ is } K_1 \text{ or } K_2, \\ 1 & \text{otherwise.} \end{cases}$$

and then deduce Theorem 4.

Proof (Proof of Theorem 3). Let G be a PURE-2-DIR graph. Without loss of generality, we can assume that segments are either horizontal or vertical, and that they are numbered from top to bottom and left to right. Hence, denoting a_1, \dots, a_p the vertices corresponding to the horizontal segments and by b_1, \dots, b_q the vertices corresponding to the vertical segments, it is easily checked that a_i is adjacent to b_j if and only if there exist $i_1 \leq i \leq i_2$ and $j_1 \leq j \leq j_2$ such that a_{i_1} and a_{i_2} are adjacent to b_j and b_{j_1} and b_{j_2} are adjacent to a_i . The result then follows from Lemma 6 (see Fig. 3 for an example of a single obstacle representation of a PURE-2-DIR graph). \square

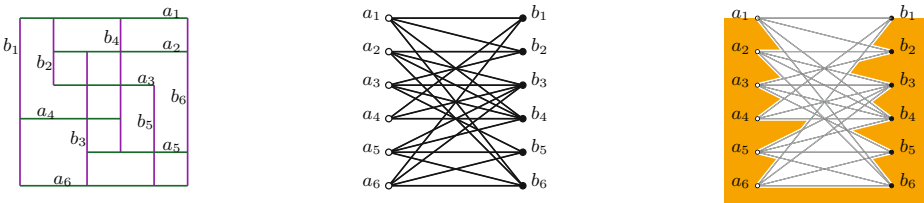


Fig. 3. Construction of a representation of a graph $G \in 2\text{-DIR}$ with a single obstacle.

Proof (Proof of Theorem 4). It has been proved in [8] that every bipartite planar graph can be represented as PURE-2-DIR graph. Thus it directly follows from Theorem 3 that the obstacle number of every planar bipartite graph different from K_1 and K_2 is 1. \square

Note that the linear time algorithm to compute a representation of a bipartite planar graph as a PURE-2-DIR graph given in [9] can be easily modified to provide a linear time algorithm providing a representation of a planar bipartite graph as the visibility graph of a family of points with a single polygonal obstacle.

Acknowledgments. We thank Vít Jelínek for ideas leading to a simplification of our proof. We also thank Daniel Král and Roman Nedela for inspiring comments on our research.

References

1. Alpert, H., Koch, C., Laison, J.D.: Obstacle numbers of graphs. *Discrete Comput. Geom.* **44**(1), 223–244 (2010)
2. Balko, M., Cibulka, J., Valtr, P.: Drawing graphs using a small number of obstacles. In: Di Giacomo, E., Lubiw, A. (eds.) *GD 2015*. LNCS, vol. 9411, pp. 360–372. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27261-0_30
3. Berman, L.W., Chappell, G.G., Faudree, J.R., Gimbel, J., Hartman, C., Williams, G.I.: Graphs with obstacle number greater than one. [arXiv:1606.03782](https://arxiv.org/abs/1606.03782) (2016)
4. Chaplick, S., Lipp, F., Park, J., Wolff, A.: Obstructing visibilities with one obstacle. [arXiv:1607.00278](https://arxiv.org/abs/1607.00278) (2016)
5. De Berg, M., Van Kreveld, M., Overmars, M., Schwarzkopf, O.C.: Computational geometry. In: *Computational Geometry*, pp. 1–17. Springer, Heidelberg (2000). https://doi.org/10.1007/978-3-540-77974-2_1
6. Dujmović, V., Morin, P.: On obstacle numbers. *Electron. J. Combin.* **22**(3), P3 (2013)
7. Fáry, I.: On straight line representation of planar graphs. *Acta Scientiarum Mathematicarum (Szeged)* **II**, 229–233 (1948)
8. de Fraysseix, H., Ossona de Mendez, P., Pach, J.: Representation of planar graphs by segments. In: *Intuitive Geometry*, vol. 63. *Colloquia Mathematica Societatis János Bolyai*, pp. 109–117. North-Holland (1991)
9. de Fraysseix, H., Ossona de Mendez, P., Pach, J.: A left-first search algorithm for planar graphs. *Discrete Comput. Geom.* **13**, 459–468 (1995)
10. de Fraysseix, H., Pach, J., Pollack, R.: Small sets supporting Fáry embeddings of planar graphs. In: *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, STOC 1988*, pp. 426–433. ACM, New York (1988)
11. de Fraysseix, H., Pach, J., Pollack, R.: How to draw a planar graph on a grid. *Combinatorica* **10**, 41–51 (1990)
12. Fulek, R., Saeedi, N., Sarióž, D.: Convex obstacle numbers of outerplanar graphs and bipartite permutation graphs. In: Pach, J. (ed.) *Thirty Essays on Geometric Graph Theory*, pp. 249–261. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-0110-0_13
13. Ghosh, S.K.: *Visibility Algorithms in the Plane*. Cambridge University Press, New York (2007)
14. Ghosh, S.K., Goswami, P.P.: Unsolved problems in visibility graphs of points, segments, and polygons. *ACM Comput. Surv. (CSUR)* **46**(2), 22 (2013)
15. Kratochvíl, J., Matoušek, J.: Intersection graphs of segments. *J. Combin. Theory Ser. B* **62**(B)(2), 289–315 (1994)
16. Mohar, B., Thomassen, C.: *Graphs on Surfaces*. The Johns Hopkins University Press, Baltimore (2001)
17. O’Rourke, J.: Visibility. In: *Handbook of Discrete and Computational Geometry*, pp. 467–479. CRC Press (1997)
18. O’Rourke, J.: Open problems in the combinatorics of visibility and illumination. In: *Advances in Discrete and Computational Geometry: Proceedings of the 1996 AMS-IMS-SIAM Joint Summer Research Conference, Discrete and Computational Geometry-Ten Years Later, 14–18 July 1996, Mount Holyoke College*, vol. 223, p. 237. American Univ. in Cairo Press (1999)

19. Pach, J., Sariöz, D.: On the structure of graphs with low obstacle number. *Graphs and Combinatorics* **27**(3), 465–473 (2011)
20. Sariöz, D.: Approximating the obstacle number for a graph drawing efficiently. In: *Proceedings of 23rd Canadian Conference on Computational Geometry*, pp. 297–302 (2011)
21. Steinitz, E.: *Polyeder und Raumeinteilungen*. Teubner (1916)
22. Tutte, W.T.: Toward a theory of crossing numbers. *J. Comb. Theory* **8**, 45–53 (1970)
23. Urrutia, J.: Art gallery and illumination problems. In: *Handbook of Computational Geometry*, pp. 973–1027. North-Holland (2000)

Grid-Obstacle Representations with Connections to Staircase Guarding

Therese Biedl^(✉) and Saeed Mehrabi

Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada
{biedl,smehrabi}@uwaterloo.ca

Abstract. In this paper, we study grid-obstacle representations of graphs where we assign grid-points to vertices and define obstacles such that an edge exists if and only if an xy -monotone grid path connects the two endpoints without hitting an obstacle or another vertex. It was previously argued that all planar graphs have a grid-obstacle representation in 2D, and all graphs have a grid-obstacle representation in 3D. In this paper, we show that such constructions are possible with significantly smaller grid-size than previously achieved. Then we study the variant where vertices are not blocking, and show that then grid-obstacle representations exist for bipartite graphs. The latter has applications in so-called *staircase guarding* of orthogonal polygons; using our grid-obstacle representations, we show that staircase guarding is NP-hard in 2D.

1 Introduction

Recently, Bishnu et al. [6] initiated the study of *grid-obstacle representations*. Here the vertices of a graph $G = (V, E)$ are mapped to points in an integer grid, and other grid-points are marked as *obstacles* in such a way that (v, w) is an edge of G if and only if there exists an xy -monotone path in the grid from v to w that contains no obstacle-point and no point that belongs to some vertex $\neq v, w$. See also Fig. 1b. This is a special case of a more general problem, which asks for placing points and obstacles in the plane such that an edge (v, w) exists if and only if there is a shortest path (in some distance metric) from v to w that does not intersect obstacles. See also Alpert et al. [2], who initiated the study of obstacle numbers, and [9] and the references therein for more recent developments.

Bishnu et al. [6] showed that any planar graph has a grid-obstacle representation in 2D, and every graph has a grid-obstacle representation in 3D. The main idea was to use a straight-line drawing, and then approximate it by putting a sufficiently fine grid around it that consists of obstacles everywhere except near the edge. The analysis of how fine a grid is required is not straightforward; Bishnu et al. claimed that in 2D an $O(n^2) \times O(n^2)$ -grid is sufficient. They did not give bounds for the size needed in 3D (but it clearly is polynomial and at least $\Omega(n^2)$ in each dimension). Pach showed that not all bipartite graphs have grid-obstacle representations in 2D [12].

In this paper, we improve the grid-size bounds of [6]. In particular, rather than converting a straight-line drawing directly into a grid-obstacle representation, we first convert it into a visibility representation or an orthogonal drawing that has special properties, but resides in a linear-size grid. This can then be easily converted to a grid-obstacle representation. Thus we obtain 2D grid-obstacle representations for planar graphs in an $O(n) \times O(n)$ -grid, and 3D grid-obstacle representations for all graphs in an $O(n) \times O(n) \times O(n)$ -grid.

We then discuss the case with the restriction that vertices act as obstacles for edges not incident to them, and show that sometimes this restriction can be dropped. We hence obtain *non-blocking grid-representations* in 2D for all planar bipartite graphs and in 3D for arbitrary bipartite graphs.

The latter has applications: we can use the constructions for hardness proofs for a polygon-guarding problem. A point guard g is said to *staircase guard* (or *s-guard* for short) a point p inside an orthogonal polygon P if p can be reached from g by a *staircase*; that is, an orthogonal path inside P that is both x - and y -monotone. In the *s-guarding problem*, the objective is to guard an orthogonal polygon with the minimum number of s -guards. Motwani et al. [11] proved that s -guarding is polynomial on simple orthogonal polygons. Gewali and Ntafos [10] proved that the problem is NP-hard in 3D; since they reduce from vertex cover in graphs with maximum degree 3 this in fact implies APX-hardness in 3D [1]. To our knowledge, however, the complexity was open for 2D polygons with holes. Using non-blocking grid-representations, we show that it is NP-hard.

2 2D Grid-Obstacle Representations

Let $G = (V, E)$ be a planar graph. To build a grid-obstacle representation, we use a *visibility representation* where every vertex is represented by a *bar* (a horizontal line segment), and every edge is represented by a vertical line segment between the bars corresponding to the endpoints of the edge [13–15]. We need here a construction with a special property, which can easily be achieved by “shifting” where the edges attach at the vertices (see the full paper for a direct proof). See also Fig. 1a.

Lemma 1. *Every planar graph has a visibility representation in an $O(n) \times O(n)$ -grid for which any vertex-bar can be split into a left and right part such that all downward edges attach on the left and all upward edges attach on the right.*

Now convert such a visibility representation into a grid-obstacle representation. First, double the grid so that no two grid-points on edge-segments or vertex-bars are adjacent unless the corresponding graph-elements were. For each vertex v , assign as vertex-point some grid-point that lies between the two parts of the bar of v ; this exists since we doubled the grid. The obstacles consist of all grid points that are not on some edge segment or vertex bar. Clearly, the representation is in an $O(n) \times O(n)$ -grid. In the full paper, we show that this is a grid-obstacle representation, and so we have:

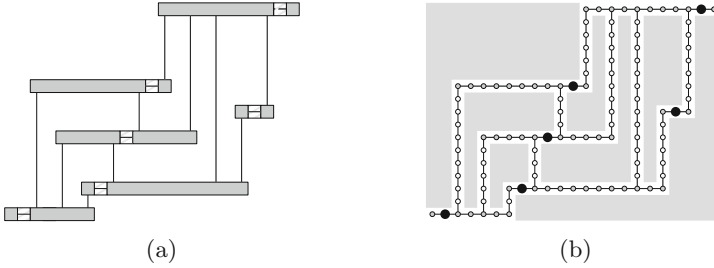


Fig. 1. A special visibility representation gives a grid-obstacle representation.

Theorem 1. *Every planar graph has a 2D grid-obstacle representation in an $O(n) \times O(n)$ -grid.*

One can easily argue that any straight-line drawing of a planar graph of height H can be converted into a visibility representation of height $2H$ and width $O(n)$ (see also [4]). Then we can apply the same approach as above. Based on drawings for trees [8], outer-planar graphs [3] and series-parallel graphs [3], we hence get:

Corollary 1. *Every tree and every outer-planar graph has a 2D grid-obstacle representation in an $O(\log n) \times O(n)$ -grid. Every series-parallel graph has a 2D grid-obstacle representation in an $O(\sqrt{n}) \times O(n)$ -grid.*

3 3D Grid-Obstacle Representation

In this section, we argue that a similar (and even simpler) construction gives a grid-obstacle representation in 3D. We obtain this by building an orthogonal representation first that has special properties. This representation is not quite a graph drawing, because edges may overlap; this will not create problems for the obstacle representation later.

Enumerate the vertices as v_1, \dots, v_n in arbitrary order. Place v_i at (i, i, i) . To draw an edge (v_i, v_j) with $i < j$, we use the path $(i, i, i) - (j, i, i) - (j, i, j) - (j, j, j)$ along the cube spanned between the two points. See Fig. 2. Observe that all edges (v_h, v_i) with $h < i$ reach v_i from the y^- -side and that all edges (v_i, v_j) with $i < j$ leave v_i at the x^+ -side. Edges incident to v_i may overlap along these two sides, but otherwise there are no overlaps or crossings in the drawing. Also, we clearly reside in an $n \times n \times n$ -grid.

Now double the grid, then cover any grid-point by an obstacle unless it is used by a vertex or an edge. One can easily argue that the result is a grid-obstacle representation (see the full paper for a formal proof), and we have:

Theorem 2. *Every graph has a 3D grid-obstacle representation in an $O(n) \times O(n) \times O(n)$ -grid.*

Notice that the obstacle in this case can be made to be just one polyhedron (albeit of high genus).

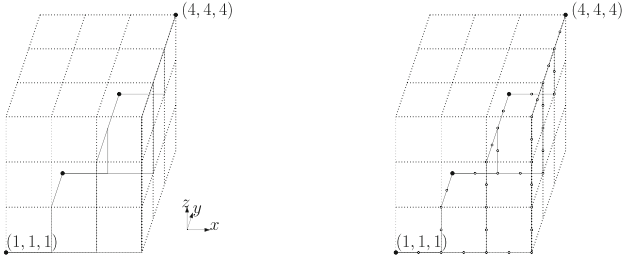


Fig. 2. A 3D orthogonal representation of K_4 , and converting it into a grid-obstacle representation. All grid-points that are not shown are blocked by obstacles.

4 Non-blocking Grid-Obstacle Representations

In our definition of grid-obstacle representation, we required that the grid point of any vertex v acts as an obstacle to any other path. The main reason for this is that otherwise paths could “seep through” a vertex, creating unwanted adjacencies. In this section, we consider *non-blocking grid-obstacle representations*, which means that vertices do not act as obstacles.

4.1 Planar Bipartite Graphs

We first give an algorithm for non-blocking grid-obstacle representation of planar bipartite graphs. It is known that any such graph $G = (A \cup B, E)$ has an *HH-drawing* [5], i.e., a planar drawing where all vertices in A have positive y -coordinate, all vertices in B have negative y -coordinate, every edge is drawn with at most one bend, and all bends have y -coordinate 0. See also Fig. 3.

In particular, we know that every edge is drawn y -monotonically. Any such drawing can be converted into a visibility representation [4] where the y -coordinate of every vertex is unchanged. So we obtain:

Lemma 2. *Let $G = (A \cup B, E)$ be a planar bipartite graph. Then, there exists a visibility representation of G such that all vertices in A have only neighbours below, and all vertices in B have only neighbours above.*

Now create an obstacle representation as before by doubling the grid, and placing obstacles at all grid-points that are not used by the drawing. Place each vertex $a \in A$ at the rightmost grid-point of the bar of a , and each $b \in B$ at the leftmost grid-point of the bar of b . One easily verifies that this is a non-blocking grid-obstacle representation: For each vertex a in A , no xy -monotone path can go through the grid-point of a without ending there, because no grid-point higher than a can be reached when going through a . Similarly one argues for B , and so we have:

Theorem 3. *Every planar bipartite graph has a non-blocking grid-obstacle representation in an $O(n) \times O(n)$ -grid.*

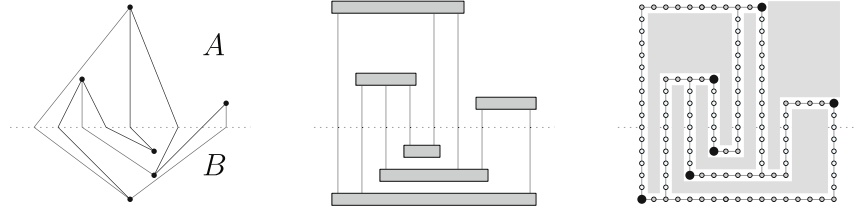


Fig. 3. An HH-drawing of a planar bipartite graph, and converting it to a non-blocking grid-obstacle representation.

4.2 Application to Staircase Guarding

Recall that the s -guarding problem consists of finding the minimum set S of points in a given orthogonal polygon P such that for any $q \in P$ there exists a $p \in S$ that is connected to q via a staircase inside P . Using non-blocking grid-obstacle representations, we can show:

Theorem 4. s -guarding is NP-hard on orthogonal polygons with holes.

Proof. We reduce from minimum dominating set, i.e., the problem of finding a set D of vertices in a graph such that every vertex is either in D or has a neighbor in D . This is NP-hard, even on planar bipartite graphs [7]. Given a planar bipartite graph $G = (A \cup B, E)$, construct the non-blocking grid-obstacle representation Γ from Theorem 3. Let P' consist of all unit squares (pixels) around grid-points that are not in an obstacle. The obstacles of Γ become holes in P' . Now for any vertex $a \in A$ extend the bar of a slightly rightward beyond the last edge, and for every $b \in B$ extend the bar leftward beyond the last edge. Finally, at every edge e , attach two “spirals” on the left and right side of its vertical segment; the one on the left curls upward while the one on the right curls downward. See Fig. 4. These spirals are small enough that they fit within the holes of P' , without overlapping other parts of P' or each other. We show in the full paper that G has a dominating set of size k if and only if this polygon can be s -guarded with $2|E| + k$ guards. This proves the theorem. \square

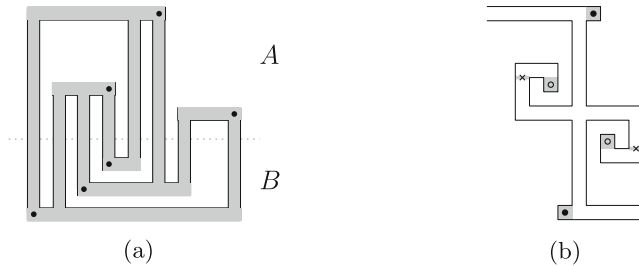


Fig. 4. The polygon for the graph in Fig. 3, and gadgets that we attach.

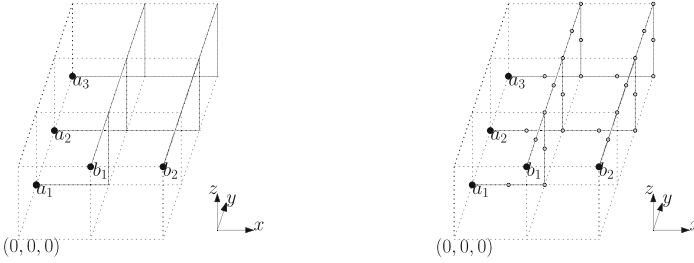


Fig. 5. A 3D orthogonal representation of $K_{2,3}$, and converting it into a grid-obstacle representation. Grid-points not shown are covered by obstacles.

4.3 3D Grid-Obstacle Representation of Bipartite Graphs

In 3D, all bipartite graphs have a non-blocking grid-obstacle representation: Enumerate the vertices as $A = \{a_1, \dots, a_\ell\}$ and $B = \{b_1, \dots, b_k\}$. Place a point for vertex a_i at $(0, i, 0)$ and a point for vertex b_j at $(j, 0, 1)$. Route each edge (a_i, b_j) as the orthogonal path $(0, i, 0) - (j, i, 0) - (j, i, 1) - (j, 0, 1)$, and observe that two paths overlap in the x^+ -direction at a_i if they both begin at a_i , or overlap in the y^+ -direction at b_j if they both end at b_j , but otherwise there is no overlap. Now obtain the grid-obstacle representation as before by doubling the grid and making grid-points obstacles unless they are used by vertices and edge-paths (Fig. 5). As before one argues that this is indeed a non-blocking grid-obstacle representation and so we have:

Theorem 5. *Every bipartite graph has a 3D non-blocking grid-obstacle representation.*

5 Conclusion

In this paper, we studied grid-obstacle representations. We gave constructions with smaller grid-size for planar graphs in 2D and all graphs in 3D. If the graph is bipartite then we can construct representations where vertices are not considered obstacles. We used these types of representation to prove NP-hardness of the s -guarding problem in 2D polygons with holes.

It remains open whether an asymptotically smaller grid and/or fewer obstacles might be enough. If we allow obstacles to be polygons rather than grid-points, we use (in Theorems 1 and 3) one obstacle per face of the planar graph, or $\Theta(n)$ in total. For grid-obstacle representations that use straight-line segments, rather than xy -monotone grid-paths, significantly fewer obstacles suffice [9]. Can we create grid-obstacle representations with $o(n)$ obstacles, at least for some subclasses of planar graphs? Another direction for future work would be to find other classes of graphs for which we can construct non-blocking grid-obstacle representations. Does this exist for all planar graphs in 2D?

References

1. Alimonti, P., Kann, V.: Some APX-completeness results for cubic graphs. *Theor. Comput. Sci.* **237**(1–2), 123–134 (2000)
2. Alpert, H., Koch, C., Laison, J.D.: Obstacle numbers of graphs. *Discrete Comput. Geom.* **44**(1), 223–244 (2010)
3. Biedl, T.: Small drawings of outerplanar graphs, series-parallel graphs, and other planar graphs. *Discrete Comput. Geom.* **45**(1), 141–160 (2011)
4. Biedl, T.: Height-preserving transformations of planar graph drawings. In: Duncan, C., Symvonis, A. (eds.) GD 2014. LNCS, vol. 8871, pp. 380–391. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45803-7_32
5. Biedl, T., Kaufmann, M., Mutzel, P.: Drawing planar partitions II: HH-drawings. In: Hromkovič, J., Sýkora, O. (eds.) WG 1998. LNCS, vol. 1517, pp. 124–136. Springer, Heidelberg (1998). https://doi.org/10.1007/10692760_11
6. Bishnu, A., Ghosh, A., Mathew, R., Mishra, G., Paul, S.: Grid obstacle representations of graphs (2017). coRR report [arXiv:1708.01765](https://arxiv.org/abs/1708.01765)
7. Clark, B.N., Colbourn, C.J., Johnson, D.S.: Unit disk graphs. *Discrete Math.* **86**(1–3), 165–177 (1990)
8. Crescenzi, P., Di Battista, G., Piperno, A.: A note on optimal area algorithms for upward drawings of binary trees. *Comput. Geom.* **2**, 187–200 (1992)
9. Dujmovic, V., Morin, P.: On obstacle numbers. *Electr. J. Comb.* **22**(3), P3.1 (2015)
10. Gewali, L., Ntafos, S.C.: Covering grids and orthogonal polygons with periscope guards. *Comput. Geom.* **2**, 309–334 (1992)
11. Motwani, R., Raghunathan, A., Saran, H.: Covering orthogonal polygons with star polygons: the perfect graph approach. *J. Comput. Syst. Sci.* **40**(1), 19–48 (1990)
12. Pach, J.: Graphs with no grid obstacle representation. *Geombinatorics* **26**(2), 80–83 (2016)
13. Rosenstiehl, P., Tarjan, R.E.: Rectilinear planar layouts and bipolar orientation of planar graphs. *Discrete Comput. Geom.* **1**, 343–353 (1986)
14. Tamassia, R., Tollis, I.: A unified approach to visibility representations of planar graphs. *Disc. Comput. Geom.* **1**, 321–341 (1986)
15. Wismath, S.: Characterizing bar line-of-sight graphs. In: ACM Symposium on Computational Geometry (SoCG 1985), pp. 147–152. ACM (1985)

Reconstructing Generalized Staircase Polygons with Uniform Step Length

Nodari Sitchinava¹ and Darren Strash²(✉)

¹ Department of Information and Computer Sciences, University of Hawaii,
Manoa, USA

`nodari@hawaii.edu`

² Department of Computer Science, Colgate University, Hamilton, NY, USA
`dstrash@cs.colgate.edu`

Abstract. *Visibility graph reconstruction*, which asks us to construct a polygon that has a given visibility graph, is a fundamental problem with unknown complexity (although visibility graph recognition is known to be in PSPACE). We show that two classes of uniform step length polygons can be reconstructed efficiently by finding and removing rectangles formed between consecutive convex boundary vertices called tabs. In particular, we give an $O(n^2m)$ -time reconstruction algorithm for orthogonally convex polygons, where n and m are the number of vertices and edges in the visibility graph, respectively. We further show that reconstructing a monotone chain of staircases (a histogram) is fixed-parameter tractable, when parameterized on the number of tabs, and polynomially solvable in time $O(n^2m)$ under reasonable alignment restrictions.

Keywords: Visibility graphs · Polygon reconstruction
Visibility graph recognition · Orthogonal polygons
Fixed-parameter tractability

1 Introduction

Visibility graphs, used to capture visibility in or between polygons, are simple but powerful tools in computational geometry. They are integral to solving many fundamental problems, such as routing in polygons, and art gallery and watchman problems, to name a few. Efficient, and even worst-case optimal, algorithms exist for computing a visibility graph from an input polygon [16]; however, comparatively little is known about the reverse direction: the so-called visibility graph *recognition* and *reconstruction* problems.

In this paper, we study *vertex-vertex visibility graphs*, which are formed by visibility between pairs vertices of a polygon. Given a graph $G = (V, E)$, the visibility graph recognition problem asks if G is the visibility graph of *some*

N. Sitchinava—This material is based upon work supported by the National Science Foundation under Grant No. 1533823.

polygon. Similarly, the visibility graph reconstruction problem asks us to construct a polygon with G as a visibility graph. Surprisingly, recognition of simple polygons is only known to be in PSPACE [13], and it is still unknown if simple polygons can be reconstructed in polynomial time. Therefore, current solutions are typically for restricted classes of polygons.

1.1 Special Classes

A well-known result due to ElGindy [11] is that every maximal outerplanar is a visibility graph and a polygon can be reconstructed from every such graph in polynomial time. Other special classes rely on a unique configuration of reflex and convex chains, which restrict visibility. For instance, spiral polygons [14], and tower polygons [7] (also called funnel polygons), can be reconstructed in linear time, and each consists of one and two reflex chains, respectively. 2-spirals can also be reconstructed in polynomial time [3], as can a more general class of visibility graphs related to 3-matroids [4].

For monotone polygons, Colley [8,9] showed that if each face of a maximal outerplanar graph is replaced by a clique on the same number of vertices, then the resulting graph is a visibility graph of some uni-monotone polygon (monotone with respect to a single edge), and such a polygon can be reconstructed if the Hamiltonian cycle of the boundary edges is known. However, not every uni-monotone polygon (even those with uniformly spaced vertices) has such a visibility graph [12]. Finally, Evans and Saeedi [12] characterized terrain visibility graphs, which consist of a single monotone polygonal line.

For orthogonal polygons, *orthogonal convex fans* (also known as *staircase polygons*), which consist of a single staircase and an extra vertex, can be recognized in polynomial time [2]; however—strikingly—the *only* class of orthogonal polygons known to be reconstructible in polynomial time is the staircase polygon with uniform step lengths, due to Abello and Eǧecioglu [1]. Other algorithms for orthogonal polygons use different visibility representations such as vertex-edge or edge-edge visibility [18, Sect. 7.3], or “stabs” [17]. See Asano et al. [5] or Ghosh [15] for a thorough review of results on visibility graphs.

1.2 Our Results

In this work, we investigate reconstructing polygons consisting of multiple uniform step length staircases. We first show that orthogonally convex polygons can be reconstructed in time $O(n^2m)$. We further show that reconstructing orthogonal uni-monotone polygons is fixed-parameter tractable, when parameterized on the number of the horizontal convex-convex boundary edges in the polygon. We also provide an $O(n^2m)$ time algorithm under reasonable alignment assumptions. As a consequence of our reconstruction technique, we can also recognize the visibility graphs of these classes of polygons with the same running times.

2 Preliminaries

Let P be a polygon on n vertices. We say that a point p *sees* a point q (or p and q are *visible*) in polygon P if the line segment pq does not intersect the exterior of P . Under this definition, visibility is allowed along edges and through vertices.

For our visibility graph discussion, we adopt standard notation for graphs and polygons. In particular, for a graph $G = (V, E)$, we denote the *neighborhood* of a vertex $v \in V$ by $N(v) = \{u \mid (v, u) \in E\}$, and denote the number of vertices and edges by $n = |V|$ and $m = |E|$, respectively. For a visibility graph $G_P = (V_P, E_P)$ of a polygon P , we call an edge in G_P that is an edge of P a *boundary edge*. Other edges (diagonals in P) are *non-boundary edges*.

Finally, critical to our proofs is the fact that a maximal clique in G_P corresponds to a maximal (in the number of vertices) convex region $R \subseteq P$ whose vertices are defined by vertices of P . A vertex v is called *simplicial* if $N(v)$ forms a clique, or equivalently v is in exactly one maximal clique. For our work here, we further adapt this definition for an edge. We say that an edge (u, v) is *1-simplicial* if $N(u) \cap N(v)$ is a clique, or equivalently (u, v) is in exactly one maximal clique¹. The intuition behind why we consider 1-simplicial edges is that, in orthogonal polygons with edges of uniform length, boundary edges between convex vertices are 1-simplicial, with the vertices of the clique forming a rectangle. (See Fig. 1.)

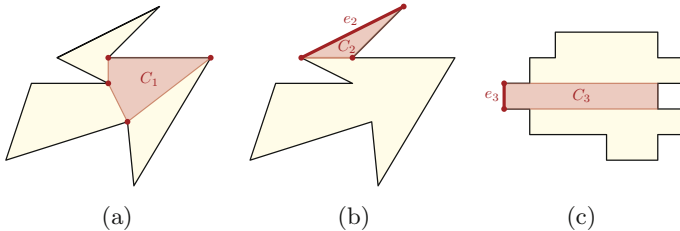


Fig. 1. Maximal convex regions on vertices of polygons are maximal cliques in visibility graphs. (b)–(c) 1-simplicial edges are in exactly one maximal clique.

Our running times depend on the following observation for 1-simplicial edges.

Observation 1. *We can test if (u, v) is 1-simplicial and in a maximal k -clique in time $O(kn)$.*

3 Uniform-Length Orthogonally Convex Polygons

We first turn our attention to a restricted class of orthogonal polygons that have only uniform-length (or equivalently, unit-length) edges. Let P be an orthogonal

¹ This is not to be confused with *simplicial edges*, which are defined elsewhere to be edges (u, v) such that for every $w \in N(u)$ and $x \in N(v)$, w and x are adjacent.

polygon with uniform-length edges such that no three consecutive vertices on P 's boundary are collinear, and further let P be *orthogonally convex*². We call P a *uniform-length orthogonally convex polygon* (UP). Note that every vertex v_i on P 's boundary is either convex or reflex. We call boundary edges between two convex vertices in a uniform-length orthogonal polygon P *tabs* and a tab's endvertices *tab vertices*. We reconstruct the polygon by computing the clockwise ordering of vertices of the UP.

Note that the boundary of a UP consists of four tabs connected via staircases. For ease of exposition, we imagine the UP embedded in \mathbb{R}^2 with polygon edges axis-aligned. We call the tab with the largest y -coordinate the north tab, and we similarly name the others the south, east, and west tabs. We similarly refer to the four boundary staircases as northwest, northeast, southeast, and southwest.

We only consider polygons with more than 12 vertices, which eliminates many special cases. Smaller polygons can be solved in constant time via brute force.

We first introduce several structural lemmas which help us identify convex vertices in a UP, which is key to our reconstruction.

Lemma 1. *For every convex vertex u in a UP there is a convex vertex v , such that $(u, v) \in E_P$ and (u, v) is 1-simplicial.*

Proof. If u is a tab vertex, then the other tab vertex v is also convex and (u, v) is 1-simplicial. Otherwise, without loss of generality, suppose that u is on the northwest staircase. Then there is a convex vertex v on the southeast staircase that is visible from u . Edge (u, v) is in exactly one maximal clique, consisting of u, v , the reflex vertices within the rectangle R defined by u and v as the opposite corners, and any other corners of R that are convex vertices of the polygon. \square

Lemma 2. *In a UP, if u or v is a reflex vertex, then edge (u, v) is not 1-simplicial.*

*Proof (Sketch*³*).* If both u and v are reflex, then (u, v) is in one maximal clique consisting of only reflex vertices and another one that includes some convex vertex w . If one of u or v is convex, there exist two convex vertices w and w' , forming two distinct maximal cliques with (u, v) . \square

Lemma 2 states that only edges between convex vertices can be 1-simplicial. Hence it allows us to identify all convex vertices, by checking for each edge (u, v) if $N(u) \cap N(v)$ is a clique in $O(n^2)$ time, leading to the following lemma.

Lemma 3. *We can identify all convex and reflex vertices in a visibility graph of a UP in $O(n^2m)$ time.*

We say a UP is *regular* if each of its staircase boundaries have the same number of vertices. Otherwise, we call it *irregular*, consisting of two *long* and two *short* staircases. We restrict our attention to irregular uniform-length orthogonally convex polygons (IUPs); however, similar methods work for their regular counterparts.

² That is, any two points in P can be connected by a staircase contained in P .

³ Full proofs may be found in the full version of this paper [19].

3.1 Irregular Uniform-Length Orthogonally Convex Polygons

Let G_P be the visibility graph of IUP P . Our reconstruction algorithm first computes the four tabs, then assigns the convex and reflex vertices to each staircase. The following structural lemma helps us find the tabs. We assume that we have already computed the convex and reflex vertices in $O(n^2m)$ time.

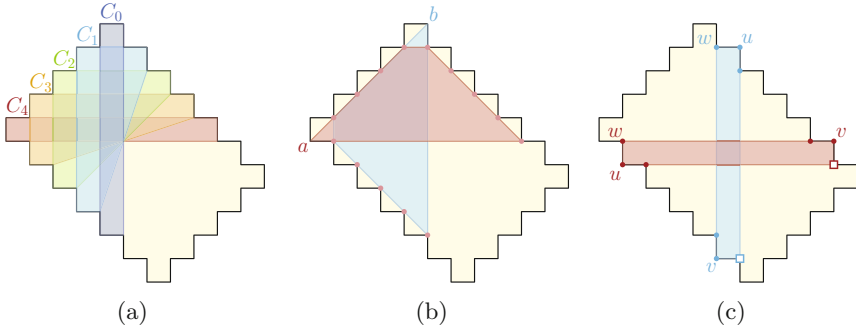


Fig. 2. Elements of our reconstruction. (a) Elementary cliques C_0, \dots, C_4 interlock along a short staircase. (b) Tab vertices a and b see unique reflex vertices on long staircases. (c) Reflex vertices (square) are discovered by forming rectangles with known vertices u, v and w .

Lemma 4. *In every IUP there are exactly four 7-vertex maximal cliques, each containing exactly three convex vertices. Each such clique contains exactly one tab, and each tab is contained in exactly one of these cliques.*

Proof. First note that each of the four tabs are in exactly one such maximal 7-clique. Further, any other clique that contains three convex vertices has at least nine vertices: each convex vertex and its two reflex boundary neighbors. \square

We note that it is not necessary to identify the four tabs explicitly to continue with the reconstruction. There are only $7^4 = O(1)$ choices of tabs (one from each 7-clique of Lemma 4), thus we can try all possible tab assignments, continue with the reconstruction and verify that our reconstruction produces a valid IUP P with the same visibility graph. However, we can explicitly find the four tabs, giving us the following lemma.

Lemma 5. *We can identify the four tabs of an IUP in $O(nm)$ time.*

We pick one tab arbitrarily to be the north tab. We conceptually orient the polygon so that the northwest staircase is short and the northeast staircase is long. We do this by computing *elementary cliques*, which identify the convex vertices on the short staircase.

Definition 1 (elementary clique). *An elementary clique in an IUP is a maximal clique that contains exactly three convex vertices: one from a short staircase, and one from each of the long staircases. (See Fig. 2(a).)*

Lemma 6. *We can identify the elementary cliques containing vertices on the northwest staircase in $O(nm)$ time.*

Proof (Sketch). Each elementary clique is constant size and contains a 1-simplicial edge, and can therefore be discovered in $O(nm)$ time. Further, elementary cliques “interlock” along a staircase: each elementary clique shares exactly three reflex vertices with its at most two neighboring elementary cliques. Thus they can be computed starting from the elementary clique containing the north tab. \square

Note that, if our sole purpose is to reconstruct the IUP P , we have sufficient information. The number of elementary cliques gives us the number of vertices on a short staircase of the polygon, from which we can build a polygon. However, in what follows, we can actually map all vertices to their positions in the IUP, which we later use to build a recognition algorithm for IUPs.

First, we show how to assign all convex vertices from the elementary cliques to each of the three staircases, using visibility of the north and west tab vertices. Note, constructing the elementary cliques with Lemma 6 also gives us the west tab, since it is contained in the last elementary clique on the northwest staircase.

Lemma 7. *We can identify the convex vertices on the northwest staircase in $O(n)$ time.*

Proof. The northwest staircase contains the convex vertices of the elementary cliques from Lemma 6 that cannot be seen by any of the north or west tab vertices. The staircase further contains the left vertex of the north tab and the top vertex of the west tab (which can be identified by the fact that they are tab vertices that do not see either vertex of the other tab). \square

We can repeat the above process to identify the convex vertices of the southeast staircase. However, we might not yet be able to identify tabs as south or east. Thus, we will obtain two possible orderings of the convex vertices on the southeast staircase. Next, we show how to assign convex vertices to the long staircases. In the process we determine south and east tabs, and consequently, identify the correct ordering of convex vertices on the southeast staircase.

Lemma 8. *We can assign the remaining convex vertices in $O(n^2)$ time.*

Proof (Sketch). Let v_i, v_{i+1} be convex vertices on the same staircase, separated by a single reflex vertex. Let u be the unique vertex on the opposite staircase, such that the angular bisector of v_i goes through u . Then u sees v_{i+1} . This likewise holds for the opposite staircase. Therefore, starting from one convex vertex on each staircase (such as a tab vertex), we can compute all convex vertices on each staircase. \square

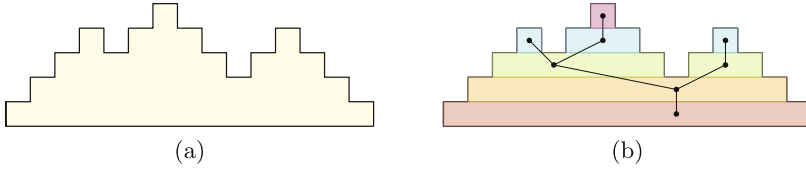


Fig. 3. (a) A histogram A with three tabs. (b) A decomposition of A into touching rectangles with a contact graph that is a tree.

Lemma 9. *We can assign the reflex vertices to each staircase in $O(n^2)$ time.*

Proof (Sketch). First we compare the reflex vertices seen by tab vertices, which gives us many vertices on the long staircases (see Fig. 2(b)). The remaining reflex vertices are discovered by building vertical and horizontal rectangles that contain unassigned reflex vertices (see Fig. 2(c)). \square

Observe that within each staircase, boundary edges are formed only between convex vertices and their reflex neighbors. Thus, we can order reflex vertices on each staircase by iterating over the staircase’s convex vertices (order of which is determined in Lemmas 6, 7 and 8) and we are done. This gives us the following result:

Theorem 1. *In $O(n^2m)$ time, we can reconstruct an IUP from its visibility graph.*

4 Uniform-Length Histogram Polygons

In this section we show how to reconstruct a more general class of uniform step length polygons: those that consist of a chain of alternating up- and down-staircases with uniform step length, which are monotone with respect to a single (longer) *base edge*. Such polygons are uniform-length *histogram polygons* [10], but we simply call them *histograms* for brevity (see Fig. 3(a) for an example). We refer to the two convex vertices comprising the base edge as *base vertices*. Furthermore, we refer to top horizontal boundary edges incident to two convex vertices as *tab edges* or just *tabs* and their incident vertices as *tab vertices*.

The Case of Two Staircases. We first note that in *double staircase* polygons (consisting of only two staircases) there is a simple linear-time reconstruction algorithm based on the degrees of vertices in the visibility graph. However, the construction relies on the symmetry of the two staircases and it is not clear whether any counting strategy works for arbitrary histograms.

4.1 Overview of the Algorithm

Every histogram can be decomposed into axis-aligned rectangles, whose contact graph is an ordered tree [10], as illustrated in Fig. 3(b). In Sect. 4.2, we show

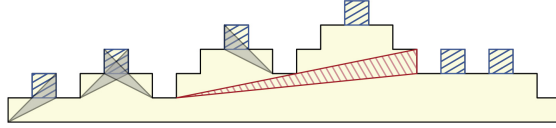


Fig. 4. Illustrating all maximal 4-cliques that contain 1-simplicial edges. These include tab cliques (square regions) and non-tab cliques (triangular regions).

that we can construct the (unordered) contact tree T from the visibility graph G_P in $O(n^2m)$ time by repeatedly “peeling” tabs from the histogram. We then show that each left-to-right ordering of T ’s k leaves (as well as a left-to-right orientation of the rectangles in the leaves) induces a histogram P' . For each candidate polygon P' (of $k!2^k$ candidates), we then compute its visibility graph $G_{P'}$ in $O(n \log n + m)$ time [16] and check if $G_{P'}$ is isomorphic to G_P . Instead of requiring an expensive graph isomorphism check [6], we show how to use the ordering of T to quickly test if G_P and $G_{P'}$ are isomorphic.

In Sect. 4.4 we show how to reduce the number of candidate histograms from $k!2^k$ to $(k-2)!2^{k-2}$, leading to the main result of our paper:

Theorem 2. *Given a visibility graph G_P of a histogram P with $k \geq 2$ tabs, we can reconstruct P in $O(n^2m + (k-2)!2^{k-2}(n \log n + m))$ time.*

Finally, we give a faster reconstruction algorithm when the histogram has a binary contact tree, solving these instances in $O(n^2m)$ time (Sect. 4.4).

4.2 Rectangular Decomposition and Contact Tree Construction

We construct the contact tree T from G_P by computing a set \mathcal{T} of the k tab edges of G_P (Lemma 11). Each tab (u, v) is 1-simplicial and in a maximal 4-clique, since $N(u) \cap N(v)$ is a 4-clique representing a unit square at the top of the histogram. Given the set \mathcal{T} of tab edges, our reconstruction algorithm picks an edge t from \mathcal{T} and removes the maximal 4-clique containing t . This is equivalent to removing an axis-aligned rectangle in P , and, equivalently, removing a leaf node from T . Moreover, it associates that node of T with four vertices of P : two *top vertices* that are convex and two *bottom vertices* that are either both reflex or are both convex *base vertices*. This process might result in a new tab edge, which we identify and add to \mathcal{T} .

Finding Initial Tabs. We start by finding the k tabs. Recall that every tab edge is 1-simplicial and in a maximal 4-clique. The converse is not necessarily true. Therefore, we begin by finding all 1-simplicial edges that are in maximal 4-cliques as a set of candidate edges, and later exclude non-tabs from the candidates.

Given a visibility graph $G_P = (V_P, E_P)$ of a histogram P and a maximal clique $C \subseteq V_P$, we call a vertex $w \in C$ an *isolated vertex with respect to P* if there exists a tab edge $(u, v) \in E_P$, such that $(N(u) \cup N(v)) \cap C = \{w\}$, i.e., of all vertices of C , only w is visible to some tab of P .

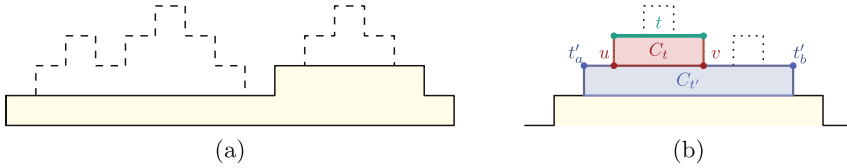


Fig. 5. (a) A truncated histogram, created by iteratively removing six tabs (dashed) from a histogram. (b) When removing C_t : t'_a, t'_b form a tab iff $t'_a, t'_b \in T \setminus C_t$, they see u and v , and $|C_{t'}| = 4$.

Lemma 10. *In a histogram, every 1-simplicial edge in a maximal 4-clique contains either a tab vertex or an isolated vertex.*

Proof (Sketch). Figure 4 shows the only types of maximal 4-cliques. ⊠

Lemma 11. *In a visibility graph of a histogram, tabs can be computed in time $O(n^2m)$.*

Proof (Sketch). We find all maximal 4-cliques in $O(nm)$ by Observation 1 and detect and eliminate those containing isolated vertices in $O(n^2m)$ time. ⊠

Note that top vertices cannot see the vertices above them. Therefore, only bottom vertices see tab vertices. Moreover, every bottom vertex sees at least one tab vertex. Thus, identifying all tabs immediately classifies vertices of G_P into top vertices and bottom vertices.

Peeling Tabs. Let P' be a polygon resulting from peeling tab cliques (rectangles) from a histogram P . We call P' a *truncated* histogram. See Fig. 5(a) for an example. After peeling a tab clique, the resulting polygon does not have uniform step length and the visibility graph may no longer have the properties on which Lemma 11 relied to detect initial tabs. Instead, we use the following lemma to detect newly created tabs during tab peeling.

Lemma 12. *When removing a tab clique from the visibility graph of a truncated histogram, any newly introduced tab can be computed in time $O(n)$.*

Proof. Denote the removed (tab) clique by C_t and let t be its tab. Let $u, v \notin t$ be the non-tab vertices of C_t . Since u sees v , (u, v) is an edge in G_P .

Since top vertices can only see vertices at and below their own level, besides the vertices of t , there are exactly two other top vertices in (remaining) G_P that see u and v , namely, the top vertices t'_a and t'_b of P on the same level as u and v (see Fig. 5(b)). Since t'_a, t'_b are adjacent in G_P , let $t' = (t'_a, t'_b)$.

When removing C_t from G_P , we can compute t'_a and t'_b in time $O(n)$ by selecting the only two top vertices adjacent to both u and v . Since t'_a and t'_b are the top vertices of a same rectangle $R_{t'}$, edge t' is 1-simplicial and is in exactly one maximal clique $C_{t'} = N(t'_a) \cap N(t'_b)$, which corresponds to the convex region $R_{t'}$. Finally, after C_t is removed, t' is a newly created tab if and only if $|C_{t'}| = 4$, which can again be tested in time $O(n)$ by computing $N(t'_a) \cap N(t'_b)$. ⊠

With each tab clique (rectangle) removal, we iteratively build the parent-child relationship between the rectangles in the contact tree T as follows. Using an array A , we maintain references to cliques being removed whose parents in T have not been identified yet. When a tab clique C_t is removed from G_P , the reference to C_t is inserted into $A[u]$, where u is one of the rectangle's bottom vertices. If the removal of C_t creates a new tab $t' = (t'_a, t'_b)$, we identify $C_{t'}$ in $O(n)$ time using Lemma 12. Recall that t' sees all bottom vertices on the same level. Thus, for every bottom vertex $u \in N(t'_a)$ (in the original graph G_P), if $A[u]$ is non-empty, we set $C_{t'}$ as the parent of the clique stored in $A[u]$ and clear $A[u]$. This takes at most $O(n)$ time for each peeling of a clique. We get the following lemma, where the time is dominated by the computation of the initial tabs:

Lemma 13. *In $O(n^2m)$ time we can construct the contact tree T of P , associate with each $v \in T$ the four vertices that define the rectangular region of v , and classify vertices of G_P as top vertices and bottom vertices.*

4.3 Mapping Candidate Polygon Vertices to the Visibility Graph

Let \hat{T} correspond to T with some left-to-right ordering of its leaves and let \hat{P} be the polygon corresponding to \hat{T} . We will map the vertices of G_P to the vertices of \hat{P} by providing for each vertex of G_P the x - and y -coordinates of a corresponding vertex of \hat{P} . Let t_1, t_2, \dots, t_k be the order of the tabs in \hat{P} . Since \hat{T} unambiguously defines the polygon \hat{P} , each node v of \hat{T} is associated with a rectangular region on the plane, and the four vertices of G_P are associated with the four corners of the rectangular region. Since by Lemma 13 every vertex of G_P is classified as a top vertex or a bottom vertex, the y -coordinate can be assigned to all vertices unambiguously, because there are two top vertices and two bottom vertices associated with each node v of \hat{T} . For every pair p, \bar{p} of top vertices or bottom vertices associated with a node in \hat{T} (we call them *companion* vertices) there is a choice of two x -coordinates: one associated with the left boundary and one associated with the right boundary of the rectangular region. Thus, determining the assignment of each top vertex and bottom vertex in G_P to the left or the right boundary is equivalent to defining x -coordinates for all vertices in G_P . Although there appears to be $2^{n/2}$ possible such assignments, there are many dependencies between the assignments due to the visibility edges in G_P . In fact, we will show that by choosing the x -coordinates of the tab vertices, we can assign all the other vertices. Thus, in what follows we consider each of the 2^k possible assignments of x -coordinates to the $2k$ tab vertices.

At times we must reason about the assignment of a vertex to the left (right) staircases associated with some tab t_j . Given \hat{T} , the x -coordinates of each vertex in the left and right staircase associated with every tab t_j is well-defined. Therefore, assigning a vertex p to a left (right) staircase of some tab t_j defines the x -coordinate of p .

In a valid histogram, companion vertices p and \bar{p} must be assigned distinct x -coordinates. Therefore, after each assignment below, we check the companion

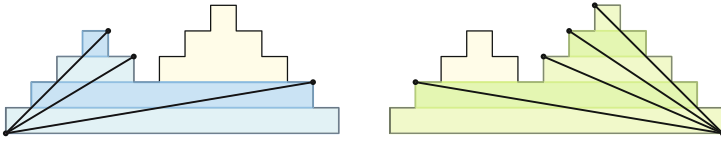


Fig. 6. Visibility from the left (right) base vertex determines the left- (right-)most tab, and orients all rectangles on the left (right) spine of the contact tree.

vertex and if they are both assigned the same x -coordinate, we exclude the current polygon candidate \hat{P} from further consideration.

We further observe that in a valid histogram, if a bottom vertex p is not in the tab clique, then it sees exactly one tab vertex, which lies on the opposite staircase associated with that tab. Thus, we assign every such bottom vertex the left (right) x -coordinate if it sees the right (left) tab vertex.

Next, consider any node v of the contact tree \hat{T} and let R_v define the rectangle associated with v in the rectangular decomposition of a valid histogram. Let p be a top vertex in R_v and let $S(p)$ be the set of vertices visible from p that are not in R_v ($S(p)$ can be determined from the neighborhood of p in G_P). Observe that if p is assigned the left (right) x -coordinate, then every vertex in $S(p)$ is a bottom vertex to the right (left) of the rectangle R_v , none of them belongs to a tab clique (i.e., all of them are already assigned x -coordinates), and all of them are assigned a right (left) x -coordinate. Since the x - and y -coordinates of the boundaries of R_v are well-defined by \hat{T} (regardless of vertex assignment), if $S(p)$ is non-empty, we check all of the above conditions and assign p an appropriate x -coordinate. If a condition is violated, then the current polygon candidate is invalid and we exclude it from further consideration.

Let p be one of the remaining top vertices without an assigned x -coordinate. If the companion \bar{p} is assigned an x -coordinate, we assign p the other choice of the x -coordinate. Otherwise, both p and \bar{p} see only the vertices inside their rectangle. In this case, the neighborhoods $N(p)$ and $N(\bar{p})$ are the same and we can assign p and \bar{p} to the opposite staircases arbitrarily.

Thus, the only remaining vertices without assigned x -coordinates are bottom vertices in tab cliques. Let R be the rectangle defined by the tab and $S_{right}(p)$ (resp., $S_{left}(p)$) denote the set of vertices that p sees among the vertices to the right (resp., left) of R . Consider a companion pair p and \bar{p} of bottom vertices that are in a tab clique. Observe that if p is on the left boundary, then $S_{right}(\bar{p}) \subseteq S_{right}(p)$ or $S_{left}(p) \subseteq S_{left}(\bar{p})$. Symmetrically, if \bar{p} is on the left boundary then $S_{right}(p) \subseteq S_{right}(\bar{p})$ or $S_{left}(\bar{p}) \subseteq S_{left}(p)$. Thus, if $|S_{right}(\bar{p})| \neq |S_{right}(p)|$ and $|S_{left}(p)| \neq |S_{left}(\bar{p})|$, we can assign p and \bar{p} appropriate x -coordinates. Otherwise, the neighborhoods $N(p)$ and $N(\bar{p})$ are the same, and we can assign p and \bar{p} to the opposite boundaries arbitrarily.

4.4 Reducing the Number of Candidate Histograms

We can reduce the number of possible orderings of tabs and staircases by considering only those that meet certain visibility constraints on the vertices that form the corners of each rectangle. In particular, we say that two rectangles $R_1 \neq R_2$ in the decomposition are *orientation-fixed* if a bottom vertex v_{bot} from one can see a top vertex v_{top} of another. Then these rectangles must be oriented so that v_{bot} and v_{top} are on opposite staircases (an up-staircase and a down-staircase). Thus, fixing an orientation of one rectangle fixes the orientation of the other.

Note that every rectangle is orientation-fixed with some leaf rectangle (as its bottom vertex can see a tab vertex). Therefore, ordering (and orienting) the leaves induces an ordering/orientation of the tree. There are $O(k!2^k)$ such orderings (and orientations) for all leaf rectangles, where k is the number of tabs.

For double staircases, T is a path and the root rectangle is orientation-fixed with every other rectangle (a base vertex is seen by every top vertex). Hence, orienting the base rectangle determines the positions of the top vertices on the double staircase. Likewise, for the histogram, the spines of T are fixed:

Lemma 14. *The base rectangle of a histogram is orientation-fixed with all rectangles on the left and right spines of T .*

Moreover, the only tab vertices visible from a base vertex are incident to the left-most or right-most tab. Thus, we can identify the left-most and right-most tabs based on the neighborhood of the base vertices. Note that removing a base rectangle of the histogram produces one or more histograms. Then we can apply this logic recursively, leading to the following algorithm:

1. Fix the orientation of the base rectangle. This identifies the rectangles on the left and right spines of T and their orientations. (See Fig. 6.)
2. The remaining subtrees collectively contain the remaining rectangles, which still must be ordered and oriented. We recursively compute the ordering and orientation of the rectangles in these subtrees.

Note if we compute the left and right spines of T , we identify the first and last tabs, and the orientations of their tab edges. Thus, we have $(k - 2)!2^{k-2}$ remaining orderings of T and orientations of the tab edges to check, as $k - 2$ tabs remain. This results in the overall reconstruction of a histogram with $k \geq 2$ tabs in $O(n^2m + (k - 2)!2^{k-2}(n \log n + m))$ time, proving Theorem 2.

We now generalize the number of orderings to consider by defining a recurrence on the tree structure. Let $v \in T$, and define $C(v)$ be v 's children in T and $d(v) = |C(v)|$. Then if we have a fixed orientation of v 's corresponding rectangle, fixing the rectangles on the left-most and right-most paths from v limits the number of possible orderings/orientations of v 's descendants to

$$F(v) \leq \begin{cases} 2^{d(v)-2} \prod_{u \in C(v)} F(u) & \text{if } d(v) > 1, \\ F(u) & \text{if } |C(v)| = 1, \text{ s.t. } C(v) = \{u\}, \\ 1 & \text{if } v \text{ is a leaf.} \end{cases}$$

Note that $F(\text{root}) = 1$ for a binary tree T . That is, the orientation of the base rectangle completely determines the histogram. Furthermore, we can find such an orientation by fixing the orientation of the base edge, determining the left- and right-most paths, ordering and orienting them to match the base edge, and then repeating this for each subtree whose root is oriented and ordered (but its children are not), which acts as a base rectangle for its subtree. This process can be done in time $O(n + m)$ by traversing T and orienting each rectangle exactly once by looking at its vertices neighbors in its base rectangle in T .

Theorem 3. *Histograms with a binary contact tree can be reconstructed in $O(n^2m)$ time.*

5 From Reconstruction to Recognition

We note that all of our reconstruction algorithms assign each vertex to a specific position in the constructed polygon. Let such an algorithm be called a *vertex assignment reconstruction*. As a result, we get recognition algorithms for these visibility graphs as well: we run our reconstruction until it fails or completes successfully, verify that the resulting polygon has the same visibility graph in time $O(n \log n + m)$ time [16], and verify that it is a polygon of the given type in linear time. Thus, we conclude that our reconstruction algorithms imply recognition algorithms with the same running times.

References

1. Abello, J., Egecioglu, Ö.: Visibility graphs of staircase polygons with uniform step length. *Int. J. Comput. Geom. Ap.* **03**(01), 27–37 (1993)
2. Abello, J., Egecioglu, Ö., Kumar, K.: Visibility graphs of staircase polygons and the weak Bruhat order, I: from visibility graphs to maximal chains. *Discrete Comput. Geom.* **14**(3), 331–358 (1995)
3. Abello, J., Kumar, K.: Visibility graphs of 2-spiral polygons (Extended abstract). In: Baeza-Yates, R., Goles, E., Poblete, P.V. (eds.) *LATIN 1995*. LNCS, vol. 911, pp. 1–15. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-59175-3_77
4. Abello, J., Kumar, K.: Visibility graphs and oriented matroids. *Discrete Comput. Geom.* **28**(4), 449–465 (2002)
5. Asano, T., Ghosh, S.K., Shermer, T.C.: Visibility in the plane. In: Urrutia, J.-R.S.J. (ed.) *Handbook of Computational Geometry*, pp. 829–876. North-Holland, Amsterdam (2000). Chap. 19
6. Babai, L.: Graph isomorphism in quasipolynomial time [extended abstract]. In: *Proceedings of 48th ACM Symposium on Theory of Computing (STOC 2016)*, pp. 684–697, New York, NY, USA. ACM (2016)
7. Choi, S.-H., Shin, S.Y., Chwa, K.-Y.: Characterizing and recognizing the visibility graph of a funnel-shaped polygon. *Algorithmica* **14**(1), 27–51 (1995)
8. Colley, P.: Visibility graphs of uni-monotone polygons. Master’s thesis, Department of Computer Science, University of Waterloo, Waterloo, Canada (1991)
9. Colley, P.: Recognizing visibility graphs of unimonotone polygons. In: *Proceedings of 4th Canadian Conference on Computational Geometry*, pp. 29–34 (1992)

10. Durocher, S., Mehrabi, S.: Computing partitions of rectilinear polygons with minimum stabbing number. In: Gudmundsson, J., Mestre, J., Viglas, T. (eds.) COCOON 2012. LNCS, vol. 7434, pp. 228–239. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32241-9_20
11. ElGindy, H.: Hierarchical decomposition of polygons with applications. Ph.D. thesis, McGill University, Montreal, Canada (1985)
12. Evans, W., Saeedi, N.: On characterizing terrain visibility graphs. *J. Comput. Geom.* **6**(1), 108–141 (2015)
13. Everett, H.: Visibility graph recognition. Ph.D. thesis, University of Toronto (1990)
14. Everett, H., Corneil, D.: Recognizing visibility graphs of spiral polygons. *J. Algorithms* **11**(1), 1–26 (1990)
15. Ghosh, S.K.: *Visibility Algorithms in the Plane*. Cambridge University Press, New York (2007)
16. Ghosh, S.K., Mount, D.M.: An output-sensitive algorithm for computing visibility. *SIAM J. Comput.* **20**(5), 888–910 (1991)
17. Jackson, L., Wismath, S.: Orthogonal polygon reconstruction from stabbing information. *Comp. Geom. Theor. Appl.* **23**(1), 69–83 (2002)
18. O’Rourke, J.: *Art Gallery Theorems and Algorithms*. Oxford University Press, New York (1987)
19. Sitchinava, N., Strash, D.: Reconstructing generalized staircase polygons with uniform step length. arXiv preprint <https://arxiv.org/abs/1708.09842> (2017)

3D Visibility Representations of 1-planar Graphs

Patrizio Angelini¹, Michael A. Bekos¹, Michael Kaufmann¹,
and Fabrizio Montecchiani²

¹ Institut für Informatik, Universität Tübingen, Tübingen, Germany
{angelini,bekos,mk}@informatik.uni-tuebingen.de

² Dipartimento di Ingegneria, Università degli Studi di Perugia, Perugia, Italy
fabrizio.montecchiani@unipg.it

Abstract. We prove that every 1-planar graph G has a z -parallel visibility representation, i.e., a 3D visibility representation in which the vertices are isothetic disjoint rectangles parallel to the xy -plane, and the edges are unobstructed z -parallel visibilities between pairs of rectangles. In addition, the constructed representation is such that there is a plane that intersects all the rectangles, and this intersection defines a bar 1-visibility representation of G .

1 Introduction

Visibility representations are a classic research topic in Graph Drawing and Computational Geometry. Motivated by VLSI applications, seminal papers studied *bar visibility representations* of planar graphs (see, e.g., [23, 26–28]), in which vertices are represented as non-overlapping horizontal segments, called *bars*, and edges correspond to vertical *visibilities* connecting pairs of bars, i.e., vertical segments that do not intersect any bar other than at their endpoints.

In order to represent non-planar graphs, more recent papers investigated models in which either two visibilities are allowed to cross, or a visibility can “go through” a vertex. Two notable examples are rectangle visibility representations and bar k -visibility representations. In a *rectangle visibility representation* of a graph, every vertex is represented as an axis-aligned rectangle and two vertices are connected by an edge using either a horizontal or a vertical visibility (see, e.g., [9, 18, 24]). A *bar k -visibility representation* is a bar visibility representation in which each visibility intersects at most k bars (see, e.g., [7, 8, 13]).

Extensions of visibility representations to 3D have also been studied. Of particular interest for us are *z -parallel visibility representations (ZPRs)*, in which the vertices of the graph are isothetic disjoint rectangles parallel to the xy -plane, and the edges are visibilities parallel to the z -axis. Bose et al. [6] proved that K_{22} admits a ZPR, while K_{56} does not. Štola [25] reduced this gap by showing that K_{51} does not admit any ZPR. If the rectangles are restricted to unit squares, then K_7 is the largest representable complete graph [14]. Other 3D visibility models are box visibility representations [15], and 2.5D box visibility representations [2].

In this paper we study 3D visibility representations of 1-planar graphs. We recall that a graph is 1-*planar* if it can be drawn with at most one crossing per

edge (see, e.g., [5, 19, 22]). The 1-planar graphs are among the most investigated families of “beyond planar graphs”, i.e., graphs that extend planarity by forbidding specific edge crossings configurations (see, e.g., [17, 21]). Brandenburg [7] and Evans et al. [13] proved that every 1-planar graph admits a bar 1-visibility representation. Later, Biedl et al. [4] proved that a 1-plane graph (i.e., an embedded 1-planar graph) admits a rectangle visibility representation if and only if it does not contain any of a set of obstructions, and that not all 1-planar graphs can be realized, regardless of their 1-planar embedding. On the other hand, every 1-planar graph can be represented with vertices that are orthogonal polygons with several reflex corners [12]. Our goal is to represent 1-planar graphs with vertices drawn as rectangles (rather than more complex polygons) by exploiting the third dimension. We prove that every 1-planar graph G has a ZPR γ . In addition, γ is 1-visible, i.e., there is a plane that is orthogonal to the rectangles of γ and such that its intersection with γ defines a bar 1-visibility representation of G (see Sect. 2 for formal definitions).

Our main contribution is summarized by the following theorem.

Theorem 1. *Every 1-planar graph G with n vertices admits a 1-visible ZPR γ in $O(n^3)$ volume. Also, if a 1-planar embedding of G is given as part of the input, then γ can be computed in $O(n)$ time.*

An embedding is needed, as recognizing 1-planar graphs is NP-complete [16, 20]. An example of a 1-visible ZPR is shown in Fig. 1. We also remark that, as pointed out by Kobourov et al. in a recent survey [19], very little is known on 3D representations of 1-planar graphs, and our result sheds some light on this problem.

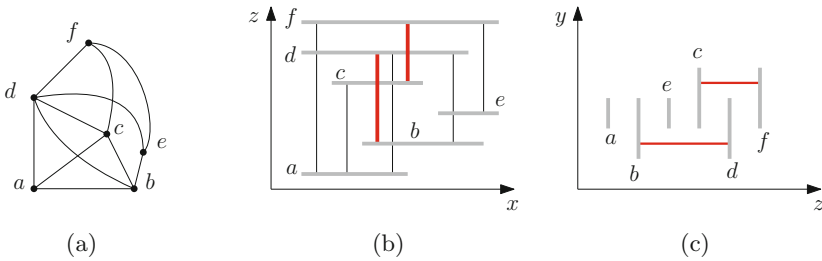


Fig. 1. (a) A 1-planar graph G . (b) The intersection of a 1-visible ZPR γ of G with the plane $Y = 0$; the red (bold) visibilities traverse a bar. (c) The projection to the yz -plane of γ (only the red visibilities are shown). (Color figure online)

From a high-level perspective, to prove Theorem 1 (see Sect. 3) we start by constructing a bar 1-visibility representation γ_1 of G , which is then used as the intersection of the ZPR γ with the plane $Y = 0$ (see, e.g., Fig. 1b). In particular, we transform each bar b of γ_1 into a rectangle R_b by computing the y -coordinates of its top and bottom sides, so that each visibility in γ_1 that traverses a bar b

can be represented as a visibility in γ that passes above or below R_b (see, e.g., Fig. 1c). This is done by using two suitable acyclic orientations of the edges of G .

For reasons of space some proofs and technicalities have been omitted and can be found in [1].

2 Preliminaries and Definitions

We assume familiarity with the concepts of planar drawings and planar embeddings, see, e.g., [10]. The *planarization* of a non-planar drawing is a planar drawing obtained by replacing every crossing with a *dummy vertex*. An *embedding* of a graph is an equivalence class of drawings whose planarized versions have the same planar embedding. A *1-plane* graph is a 1-planar graph with a *1-planar embedding*, i.e., an embedding where each edge is incident to at most one dummy vertex. A *kite* is a 1-plane graph isomorphic to K_4 in which the outer face is composed of four vertices and four crossing-free edges, while the remaining two edges cross each other. Given a 1-plane graph G and a kite $K = \{a, b, c, d\}$, with $K \subseteq G$, kite K is *empty* if it contains no vertex of G inside the 4-cycle $\langle a, b, c, d \rangle$.

A (*partial*) *orientation* \mathcal{O} of a graph G is an assignment of directions to (a subset of) the edges of G . The graph obtained by orienting the edges of G according to \mathcal{O} is the directed (or mixed) graph $G_{\mathcal{O}}$. A *planar st-(multi)graph* G is a plane acyclic directed (multi)graph with a single source s and a single sink t , with both s and t on its outer face [11]. The sets of incoming and outgoing edges incident to each vertex v of G are *bimodal*, i.e., they are contiguous in the cyclic ordering of the edges at v . Each face f of G is bounded by two directed paths with a common origin and destination, called the *left path* and *right path* of f . Face f is the *left* (resp., *right*) face for all vertices on its right (resp., left) path except for the origin and for the destination. A *topological ordering* of a directed acyclic (multi)graph is a linear ordering of its vertices such that for every directed edge from vertex u to vertex v , u precedes v in the ordering.

A set \mathcal{R} of disjoint rectangles in \mathbb{R}^3 is *z-parallel*, if each rectangle has its sides parallel to the x - and y -axis. Two rectangles of \mathcal{R} are *visible* if and only if they contain the ends of a closed cylinder C of radius $\varepsilon > 0$ parallel to the z -axis and orthogonal to the xy -plane, and that does not intersect any other rectangle.

Definition 1. A *z-parallel visibility representation* (ZPR) γ of a graph G maps the set of vertices of G to a *z-parallel set of disjoint rectangles*, such that for each edge of G the two corresponding rectangles are *visible*¹. If there is a plane that is orthogonal to the rectangles of γ and such that its intersection with γ defines a bar *k-visibility representation* of G , then γ is a *k-visible ZPR*.

¹ Our visibility model is often called *weak*, to be distinguished with the *strong* model in which visibilities and edges are in bijection. While this distinction is irrelevant when studying complete graphs (e.g., in [6, 25]), the weak model is commonly adopted to represent sparse non-planar graphs in both 2D and 3D (see, e.g., [2, 4, 7, 12, 13]).

3 Proof of Theorem 1

Let $G = (V, E)$ be a 1-plane graph with n vertices. To prove Theorem 1, we present a linear-time algorithm that takes G as input and computes a 1-visible ZPR of G in cubic volume. The algorithm works in three steps, described in the following.

Step 1. We compute a bar 1-visibility representation γ_1 of G by applying Brandenburg's linear-time algorithm [7], which produces a representation with integer coordinates on a grid of size $O(n^2)$. This algorithm consists of the following steps. (a) A 1-plane multigraph $G' = (V, E' \supseteq E)$ is computed from G such that: The four end-vertices of each pair of crossing edges of G' induce an empty kite; no edge can be added to G' without introducing crossings; if two vertices are connected by a set of $k > 1$ parallel edges, then all of them are uncrossed and non-homotopic. We remark that the embedding of G' may differ from the one of G due to the rerouting of some edges. (b) Let P be the plane multigraph obtained from G' by removing all pairs of crossing edges. Let \mathcal{O} be an orientation of P such that $P_{\mathcal{O}}$ is a planar st -multigraph. Then the algorithm by Tamassia and Tollis [26] is applied to compute a bar visibility representation of $P_{\mathcal{O}}$. (c) Finally, all pairs of crossing edges are reinserted through a postprocessing step that extends the length of some bars so to introduce new visibilities. The newly introduced visibilities traverse at most one bar each. In addition, each bar is traversed by at most one visibility.

Step 2. We transform each bar b_v of γ_1 to a preliminary rectangle R_v . We assume that γ_1 lies on the xz -plane and that the bars are parallel to the x -axis. Let $z(v)$ be the z -coordinate of b_v and let $x_L(v)$ and $x_R(v)$ be the x -coordinates of the left and right endpoints of b_v , respectively. The rectangle R_v lies on the plane parallel to the xy -plane with equation $Z = z(v)$. Also, its left and right sides have x -coordinates equal to $x_L(v)$ and $x_R(v)$, respectively. It remains to compute the y -coordinates of the top and bottom sides of R_v . We preliminarily set the y -coordinates of the bottom sides and of the top sides of all the rectangles to -1 and $+1$, respectively. All the visibilities of γ_1 that do not traverse any bar can be replaced with cylinders of radius $\varepsilon < \frac{1}{2}$. Let P' be the subgraph of G' induced by all such visibilities, and let γ_2 be the resulting ZPR. The next lemma follows.

Lemma 1. γ_2 is a ZPR of P' .

Step 3. To realize the remaining visibilities of γ_1 , we modify the y -coordinates of the rectangles. The idea is to define two partial orientations of the edges of P , denoted by \mathcal{O}_1 and \mathcal{O}_2 , to assign the final y -coordinates of the top sides and of the bottom sides of the rectangles, respectively. In particular, an edge oriented from u to v in \mathcal{O}_1 (\mathcal{O}_2) encodes that the top side (bottom side) of R_u will have y -coordinate greater (smaller) than the one of R_v . The orientations are such that if two vertices u and v see each other through a third vertex w in γ_1 , then their top (bottom) sides both have larger (smaller) y -coordinate than the one of w . Hence, both \mathcal{O}_1 and \mathcal{O}_2 are defined based on γ_1 , using the following three rules.

Let $f = \{o, u, v, d\}$ be a face of $P_{\mathcal{O}}$ (and hence of P) such that $\{o, u, v, d\}$ are part of an empty kite of G' . In what follows we assume that o is the origin and d is the destination of the face. We borrow some terminology from [7], refer to Fig. 2 (the black thin edges only). If the left (resp., right) path of f is composed of the single edge (o, d) , then f is called a *right wing* (resp., *left wing*). If both the left path and the right path of f consist of two edges, then f is a *diamond*.

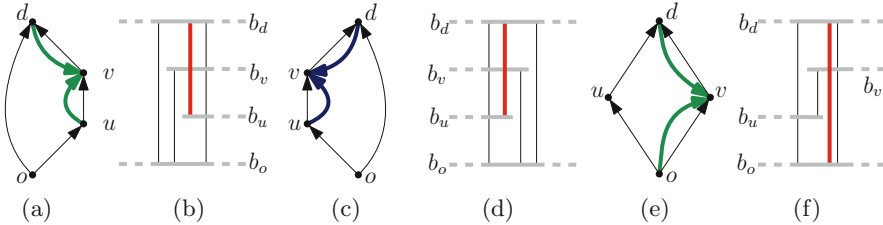


Fig. 2. (a)-(b) A right wing. (c)-(d) A left wing. (e)-(f) A diamond.

(R.1) If f is a right wing, we may assume that b_v is above b_u . Consider the restriction of γ_1 with respect to $\{o, u, v, d\}$. Either the visibility between b_u and b_d traverses b_v (as in Fig. 2b), or the visibility between b_o and b_v traverses b_u . In both cases we only orient edges in \mathcal{O}_1 . In the first case we orient (u, v) from u to v and (v, d) from d to v (see the green bold edges in Fig. 2a). In the second case we orient (o, u) from o to u and (u, v) from v to u . **(R.2)** If f is a left wing, we may assume that b_v is above b_u . As for a right wing, either the visibility between b_u and b_d traverses b_v (as in Fig. 2d), or the visibility between b_o and b_v traverses b_u . We orient the edges as for a right wing, but we only consider \mathcal{O}_2 (see, e.g., the blue bold edges in Fig. 2c). **(R.3)** If f is a diamond, we may assume that b_u is to the left of b_v . Either the visibility between b_o and b_d traverses b_v , or the visibility between b_o and b_d traverses b_u . In the first case we orient (o, v) from o to v and (v, d) from d to v in \mathcal{O}_1 (see the green bold edges in Fig. 2e). In the second case we orient (o, u) from o to u and (u, d) from d to u in \mathcal{O}_2 . By applying the above three rules for all left and right wings, and for all diamonds of $P_{\mathcal{O}}$, we obtain \mathcal{O}_1 and \mathcal{O}_2 . Note that the above procedure is correct, in the sense that no edge is assigned a direction twice. This is due to the fact that a direction in \mathcal{O}_1 (resp., \mathcal{O}_2) is assigned to an edge only if it belongs to the right (resp., left) path of a right (resp., left) wing or of a diamond. On the other hand, an edge belongs only to one right path and to one left path. In what follows, we prove that both $P_{\mathcal{O}_1}$ and $P_{\mathcal{O}_2}$ are acyclic, i.e., they have no oriented cycles.

Lemma 2. *Both $P_{\mathcal{O}_1}$ and $P_{\mathcal{O}_2}$ are acyclic.*

Sketch of proof. We prove that $P_{\mathcal{O}_1}$ is acyclic. The argument for $P_{\mathcal{O}_2}$ is symmetric. Suppose, for a contradiction, that $P_{\mathcal{O}_1}$ contains a directed cycle $C = \langle e_1, e_2, \dots, e_c \rangle$, as shown in Fig. 3a. First, note that $c > 2$. If $c = 2$, there are two non-homotopic parallel edges that are both part of the right path of a right

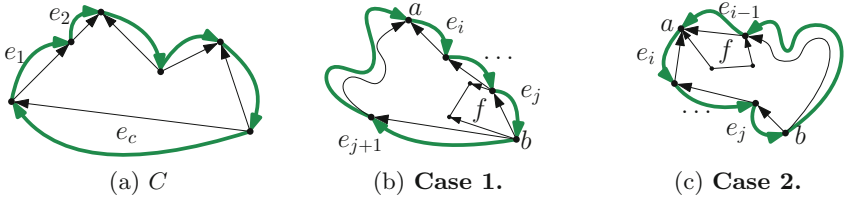


Fig. 3. Illustration for the proof of Lemma 2. In black (thin) we show the orientation of the edges according to \mathcal{O} , while in green (bold) according to \mathcal{O}_1 . (Color figure online)

wing or of a diamond in $P_{\mathcal{O}}$. But this is impossible since each pair of crossing edges in G' forms an empty kite. Some edges of C have opposite orientations in \mathcal{O} and \mathcal{O}_1 , since \mathcal{O} is acyclic. In particular, there is at least a non-empty maximal subsequence $S = \langle e_i, e_{i+1}, \dots, e_j \rangle$ of C with this property. We distinguish two cases, whether C is oriented clockwise or counter-clockwise in a closed walk along its boundary. Let a and b be the origin of e_i and the destination of e_j , respectively. Note that there is a directed path from b to a in $P_{\mathcal{O}}$ (and from a to b in $P_{\mathcal{O}_1}$).

Case 1. Refer to Fig. 3b. Since e_j is oriented in $P_{\mathcal{O}_1}$, it belongs to the right path of a right wing or of a diamond f of $P_{\mathcal{O}}$ by **R.1** and **R.3**. Also, b is the origin of f , as otherwise b would have an incoming edge between e_j and e_{j+1} in counterclockwise order from e_j , which violates the bimodality of the edges around b or the fact that the source s of $P_{\mathcal{O}}$ is on the outer face. But then the orientation of e_j in \mathcal{O}_1 contradicts **R.1** or **R.3**.

Case 2. This case can be handled similarly by observing that a is the destination of a face f having e_{i-1} in its right path. □

For each maximal subsequence of the edges of $P_{\mathcal{O}_1}$ such that each edge is oriented and the induced subgraph is connected, compute a topological ordering. Concatenate all such topological orderings, and append at the beginning or at the end of the sequence possible vertices that are not incident to any oriented edge. This gives a total ordering of the vertices of $P_{\mathcal{O}_1}$, denoted by σ_1 . Set the y -coordinate of the top side of the rectangle representing the i -th vertex in σ_1 equal to $n - i + 1$. Apply a symmetric procedure for $P_{\mathcal{O}_2}$, by computing a total ordering σ_2 , and by setting the y -coordinate of the bottom side of the rectangle representing the i -th vertex in σ_2 equal to $i - n - 1$. This concludes the construction of γ (possible dummy edges inserted by the augmentation procedure of Step 1(a) are simply ignored in γ). The correctness of γ easily follows.

Lemma 3. γ is a 1-visible ZPR of G .

Since γ_1 takes $O(n^2)$ area, and each rectangle of γ has height at most $2n$, it follows that γ takes $O(n^3)$ volume. Also, each step of the algorithm can be performed in linear time. This concludes the proof of Theorem 1.

4 Open Problems

Our research suggests interesting research directions, such as: (i) The algorithm in [7] can be adjusted to compute bar 1-visibility representations of optimal 2-planar graphs [3] (i.e., 2-planar graphs with maximum density), and our construction can be also modified to obtain 1-visible ZPRs for these graphs. Does every 2-planar graph admit a 1-visible ZPR? (ii) Can we generalize our result so to prove that every graph admitting a bar 1-visibility representation also admits a 1-visible ZPR? (iii) Our algorithm computes ZPRs in which all the rectangles are intersected by the plane $Y = 0$. Can this plane contain all bottom sides of the rectangles? If this is not possible, we wonder if every 1-planar graph admits a 2.5D-visibility representation (i.e., vertices are axis-aligned boxes whose bottom faces lie on a same plane, and visibilities are both vertical and horizontal).

References

1. Angelini, P., Bekos, M.A., Kaufmann, M., Montecchiani, F.: 3D visibility representations of 1-planar graphs. CoRR abs/1708.06196 (2017). <https://arxiv.org/abs/1708.06196>
2. Arleo, A., et al.: Visibility representations of boxes in 2.5 dimensions. In: Hu, Y., Nöllenburg, M. (eds.) GD 2016. LNCS, vol. 9801, pp. 251–265. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_20
3. Bekos, M.A., Kaufmann, M., Raftopoulou, C.N.: On optimal 2- and 3-planar graphs. In: Aronov, B., Katz, M.J. (eds.) SoCG 2017. LIPIcs, vol. 77, pp. 16:1–16:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2017)
4. Biedl, T.C., Liotta, G., Montecchiani, F.: On visibility representations of non-planar graphs. In: Fekete, S.P., Lubiw, A. (eds.) SoCG 2016. LIPIcs, vol. 51, pp. 19:1–19:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016)
5. Bodendiek, R., Schumacher, H.J., Wagner, K.: Bemerkungen zu einem Sechsfarbenproblem von G. Ringel. Abh. Math. Sem. Univ. Hamburg **53**, 41–52 (1983)
6. Bose, P., Everett, H., Fekete, S.P., Houle, M.E., Lubiw, A., Meijer, H., Romanik, K., Rote, G., Shermer, T.C., Whitesides, S., Zelle, C.: A visibility representation for graphs in three dimensions. J. Graph Algorithms Appl. **2**(3), 1–16 (1998)
7. Brandenburg, F.J.: 1-visibility representations of 1-planar graphs. J. Graph Algorithms Appl. **18**(3), 421–438 (2014)
8. Dean, A.M., Evans, W.S., Gethner, E., Laison, J.D., Safari, M.A., Trotter, W.T.: Bar k -visibility graphs. J. Graph Algorithms Appl. **11**(1), 45–59 (2007)
9. Dean, A.M., Hutchinson, J.P.: Rectangle-visibility representations of bipartite graphs. Discr. Appl. Math. **75**(1), 9–25 (1997)
10. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice-Hall, Englewood (1999)
11. Di Battista, G., Tamassia, R.: Algorithms for plane representations of acyclic digraphs. Theor. Comput. Sci. **61**, 175–198 (1988)
12. Di Giacomo, E., Didimo, W., Evans, W.S., Liotta, G., Meijer, H., Montecchiani, F., Wismath, S.K.: Ortho-polygon visibility representations of embedded graphs. In: Hu, Y., Nöllenburg, M. (eds.) GD 2016. LNCS, vol. 9801, pp. 280–294. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_22

13. Evans, W.S., Kaufmann, M., Lenhart, W., Mchedlidze, T., Wismath, S.K.: Bar 1-visibility graphs vs. other nearly planar graphs. *J. Graph Algorithms Appl.* **18**(5), 721–739 (2014)
14. Cobos, F.J., Dana, J.C., Hurtado, F., Márquez, A., Mateos, F.: On a visibility representation of graphs. In: Brandenburg, F.J. (ed.) *GD 1995. LNCS*, vol. 1027, pp. 152–161. Springer, Heidelberg (1996). <https://doi.org/10.1007/BFb0021799>
15. Fekete, S.P., Meijer, H.: Rectangle and box visibility graphs in 3D. *Int. J. Comput. Geometry Appl.* **9**(1), 1–28 (1999)
16. Grigoriev, A., Bodlaender, H.L.: Algorithms for graphs embeddable with few crossings per edge. *Algorithmica* **49**(1), 1–11 (2007)
17. Hong, S.H., Kaufmann, M., Kobourov, S.G., Pach, J.: Beyond-planar graphs: algorithmics and combinatorics (Dagstuhl Seminar 16452). *Dagstuhl Rep.* **6**(11), 35–62 (2017)
18. Hutchinson, J.P., Shermer, T.C., Vince, A.: On representations of some thickness two graphs. *Comput. Geom.* **13**(3), 161–171 (1999)
19. Kobourov, S.G., Liotta, G., Montecchiani, F.: An annotated bibliography on 1-planarity. *Comput. Sci. Rev.* **25**, 49–67 (2017). <https://doi.org/10.1016/j.cosrev.2017.06.002>
20. Korzhik, V.P., Mohar, B.: Minimal obstructions for 1-immersions and hardness of 1-planarity testing. *J. Graph Theor.* **72**(1), 30–71 (2013)
21. Liotta, G.: Graph drawing beyond planarity: some results and open problems. In: *ICTCS 2014. CEUR Workshop Proceedings*, vol. 1231, pp. 3–8. CEUR-WS.org (2014)
22. Pach, J., Tóth, G.: Graphs drawn with few crossings per edge. *Combinatorica* **17**(3), 427–439 (1997)
23. Rosenstiehl, P., Tarjan, R.E.: Rectilinear planar layouts and bipolar orientations of planar graphs. *Discr. Comput. Geom.* **1**, 343–353 (1986)
24. Shermer, T.C.: On rectangle visibility graphs. III. External visibility and complexity. In: Fiala, F., Kranakis, E., Sack, J. (eds.) *CCCG 1996*, pp. 234–239. Carleton University Press, Ottawa (1996)
25. Štola, J.: Unimaximal sequences of pairs in rectangle visibility drawing. In: Tollis, I.G., Patrignani, M. (eds.) *GD 2008. LNCS*, vol. 5417, pp. 61–66. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00219-9_7
26. Tamassia, R., Tollis, I.G.: A unified approach a visibility representation of planar graphs. *Discr. Comput. Geom.* **1**, 321–341 (1986)
27. Thomassen, C.: Plane representations of graphs. In: *Progress in Graph Theory*, pp. 43–69. AP (1984)
28. Wismath, S.K.: Characterizing bar line-of-sight graphs. In: O’Rourke, J. (ed.) *SoCG 1985*, pp. 147–152. ACM (1985)

Topological Graph Theory

Lombardi Drawings of Knots and Links

Philipp Kindermann¹(✉), Stephen Kobourov², Maarten Löffler³,
Martin Nöllenburg⁴, André Schulz¹, and Birgit Vogtenhuber⁵

¹ FernUniversität in Hagen, Hagen, Germany
{Philipp.Kindermann,Andre.Schulz}@fernuni-hagen.de

² University of Arizona, Tucson, AZ, USA
kobourov@cs.arizona.edu

³ Universiteit Utrecht, Utrecht, The Netherlands
m.loffler@uu.nl

⁴ TU Wien, Vienna, Austria
noellenburg@ac.tuwien.ac.at

⁵ Graz University of Technology, Graz, Austria
bvogt@ist.tugraz.at

Abstract. Knot and link diagrams are projections of one or more 3-dimensional simple closed curves into \mathbb{R}^2 , such that no more than two points project to the same point in \mathbb{R}^2 . These diagrams are drawings of 4-regular plane multigraphs. Knots are typically smooth curves in \mathbb{R}^3 , so their projections should be smooth curves in \mathbb{R}^2 with good continuity and large crossing angles: exactly the properties of Lombardi graph drawings (defined by circular-arc edges and perfect angular resolution).

We show that several knots do not allow plane Lombardi drawings. On the other hand, we identify a large class of 4-regular plane multigraphs that do have Lombardi drawings. We then study two relaxations of Lombardi drawings and show that every knot admits a plane 2-Lombardi drawing (where edges are composed of two circular arcs). Further, every knot is *near-Lombardi*, that is, it can be drawn as Lombardi drawing when relaxing the angular resolution requirement by an arbitrary small angular offset ε , while maintaining a 180° angle between opposite edges.

1 Introduction

A *knot* is an embedding of a simple closed curve in 3-dimensional Euclidean space \mathbb{R}^3 . Similarly, a *link* is an embedding of a collection of simple closed curves in \mathbb{R}^3 . A *drawing of a knot (link)* (also known as *knot diagram*) is a projection of the knot (link) to the Euclidean plane \mathbb{R}^2 such that for any point of \mathbb{R}^2 , at most two points of the curve(s) are mapped to it [6, 15, 16]. From a graph drawing perspective, drawings of knots and links are drawings of 4-regular plane multigraphs that contain neither loops nor cut vertices. Likewise, every 4-regular plane multigraph without loops and cut vertices can be interpreted as a link. Unless specified otherwise, we assume that a multigraph has no self-loops or cut vertices.



Fig. 1. Hand-made drawings of knots from the books of Rolfsen [15] (left), Livingston [14] (middle), and Kauffman [11] (right).

In this paper, we address a question that was recently posed by Benjamin Burton: “Given a drawing of a knot, how can it be redrawn *nice*ly without changing the given topology of the drawing?” We do know what a drawing of a knot is, but what is meant by a *nice* drawing? Several graphical annotations of knots and links as graphs have been proposed in the knot theory literature, but most of the illustrations are hand-drawn; see Fig. 1. When studying these drawings, a few desirable features become apparent: (i) edges are typically drawn as smooth curves, (ii) the angular resolution of the underlying 4-regular graph is close to 90° , and (iii) the drawing preserves the continuity of the knot, that is, in every vertex of the underlying graph, opposite edges have a common tangent.

There already exists a graph drawing style that fulfills the requirements above: a *Lombardi drawing* of a (multi-)graph $G = (V, E)$ is a drawing of G in the Euclidean plane with the following properties:

1. The vertices are represented as distinct points in the plane.
2. The edges are represented as circular arcs connecting the representations of their end vertices (and not containing the representation of any other vertex); note that a straight-line segment is a circular arc with radius infinity.
3. Every vertex has *perfect angular resolution*, i.e., its incident edges are equiangularly spaced. For knots and links this means that the angle between any two consecutive edges is 90° .

A Lombardi drawing is plane if none of its edges intersect. Note that we are particularly interested in plane Lombardi drawings, since crossings change the topology of the drawn knot.

Knot drawing software: Software for generating drawings for knots and links exists. One powerful package is `KnotPlot` [16], which provides several methods for drawing knot diagrams. It contains a library of over 1,000 precomputed knots and can also generate knot drawings of certain families, such as torus knots. `KnotPlot` also provides methods for drawing general knots based on the embedding of the underlying plane multigraph. By replacing every vertex by a 4-cycle, the multigraph becomes a simple planar 3-connected graph, which is then drawn using Tutte’s barycentric method [18]. In the end, the modifications are reversed and a drawing of the knot is obtained with edges drawn as polygonal arcs. The author noticed that this method “... does not yield ‘pleasing’ graphs or knot diagrams.” In particular, he noticed issues with vertex and angular resolution [16, pg. 102]. Another approach was used by Emily Redelmeier [1].

Here, every arc, crossing, and face of the knot diagram is associated with a disk. The drawing is then generated from the implied circle packing as a circular arc drawing. As a result of the construction, every edge in the diagram is made of three circular arcs with common tangents at opposite edges. Since no further details are given, it is hard to evaluate the effectiveness of this approach, although as we show in this paper, three circular arcs per edge are never needed. A related drawing style for knots are the so-called *arc presentations* [5]. An arc presentation is an orthogonal drawing, that is, all edges are sequences of horizontal and vertical segments, with the additional properties that at each vertex the vertical segments are above the horizontal segments in the corresponding knot and that each row and column contains exactly one horizontal and vertical segment, respectively. However, these drawings might require a large number of bends per edge.

Lombardi drawings: Lombardi drawings were introduced by Duncan et al. [8]. They showed that 2-degenerate graphs have Lombardi drawings and that all d -regular graphs, with $d \not\equiv 2 \pmod{4}$, have Lombardi drawings with all vertices placed along a common circle. Neither of these results, however, is guaranteed to result in plane drawings. Duncan et al. [8] also showed that there exist planar graphs that do not have plane Lombardi drawings, but restricted graph classes (e.g., Halin graphs) do. In subsequent work, Eppstein [9,10] showed that every (simple) planar graph with maximum degree three has a plane Lombardi drawing. Further, he showed that a certain class of 4-regular planar graphs (the medial graphs of polyhedral graphs) also admit plane Lombardi drawings and he presented an example of a 4-regular planar graph that does not have a plane Lombardi drawing. A generalization of Lombardi drawings are *k-Lombardi drawings*. Here, every edge is a sequence of at most k circular arcs that meet at a common tangent. Duncan et al. [7] showed that every planar graph has a plane 3-Lombardi drawing. Related to k -Lombardi-drawings are *smooth-orthogonal drawings of complexity k* [4]. These are plane drawings where every edge consists of a sequence of at most k quarter-circles and axis-aligned segments that meet smoothly, edges are axis-aligned (emanate from a vertex either horizontally or vertically), and no two edges emanate in the same direction. Note that in the special case of 4-regular graphs, smooth-orthogonal drawings of complexity k are also plane k -Lombardi drawings.

Our Contributions: The main question we study here is motivated by the application of the Lombardi drawing style to knot and link drawings: Given a 4-regular plane multigraph G without loops and cut vertices, does G admit a plane Lombardi drawing with the same combinatorial embedding? In Sect. 2 we start with some positive results on extending a plane Lombardi drawing, as well as composing two plane Lombardi drawings. In Sect. 3, by extending the results of Eppstein [9,10], we show that a large class of multigraphs, including 4-regular polyhedral graphs, does have plane Lombardi drawings. Unfortunately, there exist several small knots that do not have a plane Lombardi drawing. Section 4 discusses these cases but also lists a few positive results for small examples.

In Sect. 5, we show that every 4-regular plane multigraph has a plane 2-Lombardi drawing. In Sect. 6, we show that every 4-regular plane multigraph can be drawn with non-crossing circular arcs, so that the perfect angular resolution criterion is violated only by an arbitrarily small value ε , while maintaining that opposite edges have common tangents.

2 General Observations

If a knot or a link has an embedding with minimum number of vertices that admits a plane Lombardi drawing, we call it a *plane Lombardi knot (link)*. We further call the property of admitting a plane Lombardi drawing *plane Lombardiness*. If two vertices in a plane Lombardi drawing of a knot are connected by a pair of multi-edges, we denote the face enclosed by these two edges as a *lens*.

There exist a number of operations that maintain the plane Lombardiness of a 4-regular multigraph. Two knots A and B can be combined by cutting one edge of each of them open and gluing pairwise the loose ends of A with the loose ends of B . This operation is known as a *knot sum* $A + B$. Knots that cannot be decomposed into a sum of two smaller knots are known as *prime knots*. By Schubert's theorem, every knot can be uniquely decomposed into prime knots [17]. The smallest prime knot is the trefoil knot with three crossings or vertices; see Fig. 1 (right). Rolfsen's knot table¹ lists all prime knots with up to ten vertices. The Alexander-Briggs-Rolfsen notation [3, 15] is a well established notation that organizes these knots by their vertex number and a counting index, e.g., the trefoil knot 3_1 is listed as the first (and only) knot with three vertices.

Theorem 1. *Let A and B be two 4-regular multigraphs with plane Lombardi drawings. Let a be an edge of A and b an edge of B . Then the knot sum $A + B$, obtained by connecting A and B along edges a and b , admits a plane Lombardi drawing.*

Sketch of Proof. The idea of the composition is sketched in Fig. 2. We first apply Möbius transformations, rotations, and translations to the two drawings so that edges a and b can be aligned along a circle with infinite radius. This can be done such that the drawings of A and B do not intersect after removing a and b . We can now reconnect the two drawings into a single plane Lombardi drawing by introducing two edges c and d along the straight line.

Another operation that preserves the plane Lombardiness is *lens multiplication*. Let $G = (V, E)$ be a 4-regular plane multigraph with a lens between two vertices u and v . A lens multiplication of G is a 4-regular plane multigraph that is obtained by replacing the lens between u and v with a chain of lenses.

Lemma 1. *Let $G = (V, E)$ be a 4-regular plane multigraph with a plane Lombardi drawing Γ . Then, any lens multiplication G' of G also admits a plane Lombardi drawing.*

¹ http://katlas.org/wiki/The_Rolfsen_Knot_Table.

Sketch of Proof. Let f be a lens in Γ spanned by two vertices u and v as shown in Fig. 3. We draw a bisecting circular arc b that splits the angles at u and v into two 45° angles. Now we can draw any chain of lenses inside f by placing the additional vertices on b . The resulting drawing is a plane Lombardi drawing.

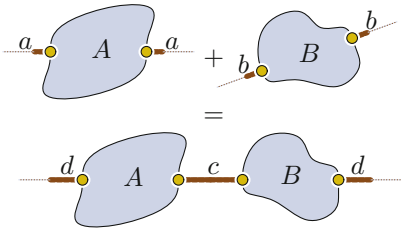


Fig. 2. Adding two plane Lombardi drawings of 4-regular multigraphs.

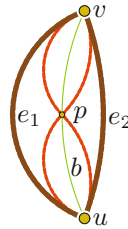


Fig. 3. Subdividing a lens between u and v for neighbor w of u and v by a new vertex p .

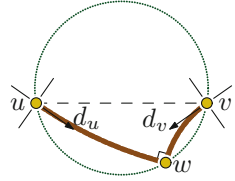


Fig. 4. Placement circle for neighbor w of u and v in a 4-regular graph.

We will use the following property several times throughout the paper.

Property 1 (Property 2 in [7,8]). Let u and v be two vertices with given positions that have a common, unplaced neighbor w . Let d_u and d_v be two tangent directions and let θ be a target angle. Let C be the locus of all positions for placing w so that (i) the edge (u, w) is a circular arc leaving u in direction d_u , (ii) the edge (v, w) is a circular arc leaving v in direction d_v , and (iii) the angle formed at w is θ . Then C is a circle, the so-called *placement circle* of w .

Duncan et al. [7] further specify the radius and center of the placement circle by the input coordinates and angles. For the special case that the two tangent directions d_u and d_v are symmetric with respect to the line through u and v , and that the angle θ is 90° or 270° , the corresponding placement circle is such that its tangent lines at u and v form an angle of 45° with the arc directions d_u and d_v . In particular, the placement circle bisects the right angle between d_u (resp. d_v) and its neighboring arc direction. Figure 4 illustrates this situation.

3 Plane Lombardi Drawings via Circle Packing

Recall that *polyhedral graphs* are simple planar 3-connected graphs, and that those graphs have a unique (plane) combinatorial embedding. The (plane) *dual graph* M' of a plane graph M has a vertex for every face of M and an edge between two vertices for every edge shared by the corresponding faces in M . In the “classic” drawing $D(M, M')$ of a primal-dual graph pair (M, M') , every vertex of M' lies in its corresponding face of M and vice versa, and every edge of M' intersects exactly its corresponding edge of M . Hence, every cell of $D(M, M')$

has exactly two such edge crossings and exactly one vertex of each of M and M' on its boundary. The *medial graph* of a primal-dual graph pair (M, M') has a vertex for every crossing edge pair in $D(M, M')$ and an edge between two vertices whenever they share a cell in $D(M, M')$; see Fig. 5a. Every cell of the medial graph contains either a vertex of M or a vertex of M' and every edge in the medial graph is incident to exactly one cell in $D(M, M')$.

Every 4-regular plane multigraph G can be interpreted as the medial graph of some plane graph M and its dual M' , where both graphs possibly contain multi-edges. If G contains no loops and cut vertices, then neither M nor M' contains loops. Eppstein [9] showed that if M (and hence also M') is polyhedral, then G admits a plane Lombardi drawing. We show next how to extend this result to a larger graph class. We only give a short sketch of the proof here. The full proof, as well as an example of the algorithm can be found in the arXiv version [12].

Theorem 2. *Let $G = (V, E)$ be a biconnected 4-regular plane multigraph and let M and M' be the primal-dual multigraph pair for which G is the medial graph. If one of M and M' is simple, then G admits a plane Lombardi drawing preserving its embedding.*

Sketch of Proof. Assume w.l.o.g. that M is simple. If M (and hence also M') is polyhedral, then G admits a plane Lombardi drawing Γ by Eppstein [9]. Moreover, Γ is embedding-preserving (up to Möbius transformation), as the combinatorial embedding of M is unique (up to homeomorphism on the sphere).

If M is not 3-connected, we proceed in three steps. First, we augment M to a polyhedral graph M_p by iteratively adding p edges (any added edge splits a face of size at least four into two faces of size at least three). During this process, we also iteratively modify the dual graph and the medial graph as shown in Fig. 5a–b.

Second, we apply Eppstein’s result to obtain a primal-dual circle packing of M_p and M'_p , together with a Lombardi drawing Γ_p of the medial graph G_p ; see Fig. 5c. Finally, we revert the augmentation process from the first step by iteratively changing the plane Lombardi drawing Γ_p of G_p to a plane Lombardi drawing Γ of G . A main ingredient for this last step is the following: In the primal-dual circle packing of M_p and M'_p , every edge g of Γ_p lies in a region $\ell(g)$ that is bounded by a primal and a dual circle. This region $\ell(g)$ is interior-disjoint with the region $\ell(g')$ of any other edge g' of Γ_p . When removing an edge from the primal graph, a vertex is removed from the Lombardi drawing and the four incident edges are replaced by two edges connecting the non-common endpoints of the four edges. We show how to draw each new edge g that replaces g_1 and g_2 in a way that again has a uniquely assigned region $\ell(g)$ that is interior-disjoint with the regions of all other edges; see Fig. 6 for a sketch.

We remark that this result is not tight: there exist 4-regular plane multigraphs whose primal-dual pair M and M' contain parallel edges that still admit plane Lombardi drawings, e.g., knots 8_{12} , 8_{14} , 8_{15} , 8_{16} ; see the arXiv version [12] for illustrations.

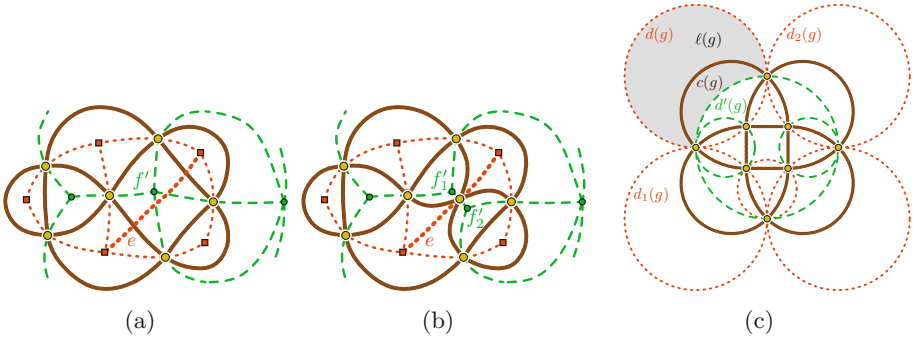


Fig. 5. (a)–(b) Modifications due to an edge addition and (c) a primal-dual circle packing. The medial graph G is drawn solid, the primal M is drawn dotted, and the dual M' is drawn dashed. The shaded area is the lens region $\ell(g)$.

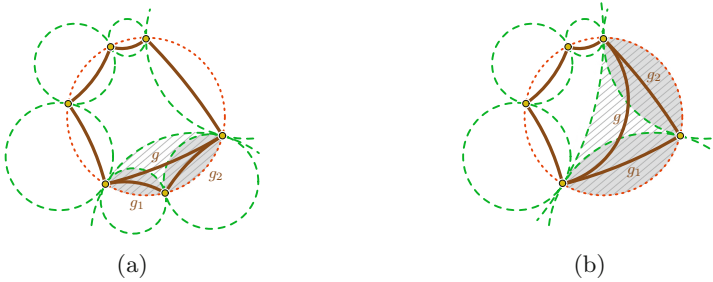


Fig. 6. Two examples of a lens region $\ell(g)$ resulting from $\ell(g_1)$ and $\ell(g_2)$: (a) convex and (b) reflex. The lens regions of g_1 and g_2 are drawn as shaded areas, while the one of g is the cross-hatched region.

We now prove that 4-regular polyhedral graphs are medial graphs of a simple primal-dual pair.

Lemma 2. *Let $G = (V, E)$ be a 4-regular polyhedral graph and let M and M' be the primal-dual pair for which G is the medial graph. If there is a multi-edge in M or in M' , then the corresponding vertices of G either have a multi-edge between them or they form a separation pair of G .*

Proof. W.l.o.g., assume that there are two edges between vertices f and g in M . Let u and v be the vertices of G that these two edges pass through; see Fig. 7. The vertices f and g of M correspond to faces in the embedding of G that both contain u and v . Hence, the removal of u and v from G disconnects G into two parts: the part inside the area spanned by the two edges between f and g and the part outside this area. Both u and v have two edges in both areas, so either

there is a multi-edge between u and v , or there are vertices in both parts, which makes u, v a separation pair of G .

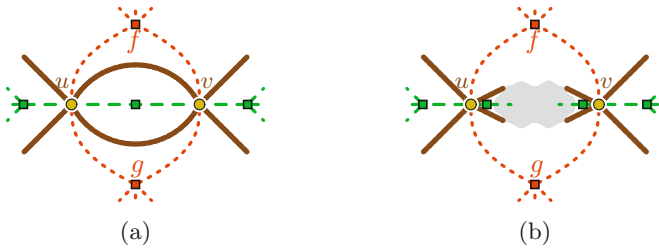


Fig. 7. If there is a multi-edge between vertices f and g in the primal, then there is a multi-edge (u, v) or a separation pair u, v in the medial.

Lemma 2 and Theorem 2 immediately give the following theorem.

Theorem 3. *Let $G = (V, E)$ be a 4-regular polyhedral graph. Then G admits a plane Lombardi drawing.*

4 Positive and Negative Results for Small Graphs

We next consider all prime knots with 8 vertices or less. We compute plane Lombardi drawings for those that have it and argue that such drawings do not exist for the others. We start by showing that no knot with a K_4 subgraph is plane Lombardi.

Lemma 3. *Every 4-regular plane multigraph G that contains K_4 as a subgraph does not admit a plane Lombardi drawing.*

Proof. Let a, b, c, d be the vertices of the K_4 . Every plane embedding of K_4 has a vertex that lies inside the cycle through the other 3 vertices; let d be this vertex. Since d has degree 4, it has another edge to either one of a, b, c , or to a different vertex. In the former case, assume that there is a multi-edge between c and d . In the latter case, by 4-regularity, there has to be another vertex of a, b, c that is connected to a vertex inside the cycle through a, b, c ; let c be this vertex. In both cases, c has two edges that lie inside the cycle through a, b, c .

Assume that G has a Lombardi drawing. Since Möbius transformations do not change the properties of a Lombardi drawing, we may assume that the edge (a, b) is drawn as a straight-line segment as in Fig. 8b. Since both c and d are neighbors of a and b , there are two corresponding placement circles by Property 1. In fact, since any two edges of a Lombardi drawing of a 4-regular graph must enclose an angle of 90° and since a and b have “aligned tangents” due to being neighbors themselves, the two placement circles coincide and a situation as shown in Fig. 8 arises. In particular, this means that in any Lombardi drawing of G the four

vertices must be co-circular. It is easy to see that we cannot draw the missing circular arcs connecting c and d : any such arc must either lie completely inside or completely outside of the placement circle. Yet, the stubs for the two edges between c and d point inside at c and outside at d .

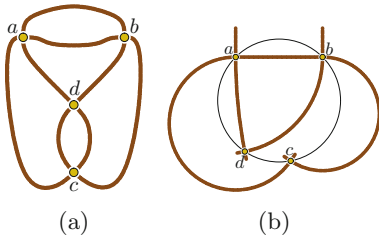


Fig. 8. Knot 4_1 (left) has no Lombardi drawing.

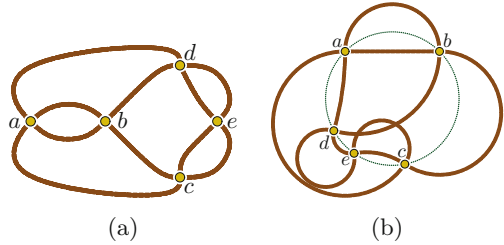


Fig. 9. Knot 5_2 and a non-plane Lombardi drawing.

The full proof for the following lemma is given in in the arXiv version [12].

Lemma 4. *Knots 4_1 and 5_2 are not plane Lombardi knots.*

Sketch of Proof. For knot 4_1 , the claim immediately follows from Lemma 3. For knot 5_2 (see Fig. 9) we can argue that all five vertices must be co-circular. Unlike knot 4_1 we can geometrically draw all edges of knot 5_2 as Lombardi arcs, see the non-plane drawing in Fig. 9b. However, by carefully considering the smooth chain of arcs $a - c - e - d - b$ and their radii, we can prove that this path must self-intersect in any Lombardi drawing, so the claim follows.

As the above lemma shows, even very small knots may not have a plane Lombardi drawings. However, most knots with a small number of crossings are indeed plane Lombardi. In the arXiv version [12], we provide plane Lombardi drawings of all knots with up to eight vertices except 4_1 and 5_2 . Most of these drawings can actually be obtained using the techniques from Sect. 2 and 3.

Theorem 4. *All prime knots with up to eight vertices other than 4_1 and 5_2 are plane Lombardi knots.*

Note that Theorem 4 implies that each of these knots has a combinatorial embedding that supports a plane Lombardi drawing. It is not true, however, that any embedding admits a plane Lombardi drawing. In fact, the knot 7_5 has an embedding that cannot be drawn plane Lombardi; details can be found in the arXiv version [12].

Theorem 5. *There exists an infinite family of prime knots and links that have embeddings that do not support plane Lombardi drawings.*

5 Plane 2-Lombardi Drawings of Knots and Links

Since not every knot admits a plane Lombardi drawing, we now consider plane 2-Lombardi drawings; see Fig. 10a for an example. Bekos et al [4] recently introduced *smooth orthogonal drawings of complexity k* . These are drawings where every edge consists of a sequence of at most k circular arcs and axis-aligned segments that meet smoothly with horizontal or vertical tangents, and where at every vertex, each edge emanates either horizontally or vertically and no two edges emanate in the same direction. For the special case of 4-regular graphs, every smooth orthogonal drawing of complexity k is also a plane k -Lombardi drawing. Alam et al. [2] showed that every plane graph with maximum degree 4 can be redrawn as a plane smooth-orthogonal drawing of complexity 2. Their algorithm takes as input an orthogonal drawing produced by the algorithm by Liu et al. [13] and transforms it into a smooth orthogonal drawing of complexity 2. We show how to modify the algorithm by Liu et al., to compute an orthogonal drawing for a 4-regular plane multigraph and then use the algorithm by Alam et al. to transform it into a smooth orthogonal drawing of complexity 2. Details are given in the arXiv version [12].

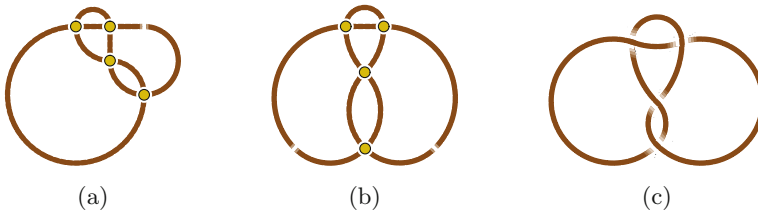


Fig. 10. Drawings of knot 4_1 which by Lemma 4 does not admit a plane Lombardi drawing. (a) A smooth orthogonal drawing of complexity 2, (b) a different plane 2-Lombardi drawing, and (c) a plane ε -angle Lombardi drawing.

Theorem 6. *Every biconnected 4-regular plane multigraph G admits a plane 2-Lombardi drawing with the same embedding.*

6 Plane Near-Lombardi Drawings

Since not all knots admit a plane Lombardi drawing, in this section we relax the perfect angular resolution constraint. We say that a knot (or a link) is *near-Lombardi* if it admits a drawing for every $\varepsilon > 0$ such that

1. All edges are circular arcs,
2. Opposite edges at a vertex are tangent;
3. The angle between crossing pairs at each vertex is at least $90^\circ - \varepsilon$.

We call such a drawing a ε -angle Lombardi drawing. Note that a Lombardi drawing is essentially a 0-angle Lombardi drawing. For example, the knot 4_1 does not admit a plane Lombardi drawing, but it admits a plane ε -angle Lombardi drawing, as depicted in Fig. 10c.

Let Γ be an ε -angle Lombardi drawing of a 4-regular graph. If each angle described by the tangents of adjacent circular arcs at a vertex in Γ is exactly $90^\circ + \varepsilon$ or $90^\circ - \varepsilon$, then we call Γ an ε -regular Lombardi drawing. Note that any Lombardi drawing is a 0-regular Lombardi drawing.

We first extend some of our results for plane Lombardi drawings to plane ε -angle Lombardi drawings. The following Lemma is a stronger version of Theorem 2. The full proof is given in the arXiv version [12].

Lemma 5. *Let $G = (V, E)$ be a biconnected 4-regular plane multigraph and let M and M' be the primal-dual multigraph pair for which G is the medial graph. If one of M and M' is simple, then G admits a plane ε -regular Lombardi drawing preserving its embedding for every $0^\circ \leq \varepsilon < 90^\circ$.*

Sketch of Proof. We first direct the edges such that every vertex has two incoming opposite edges and two outgoing opposite edges by orienting the edges around each face that belongs to M in counter-clockwise order. We use the same primal-dual circle packing approach as in Theorem 2 to obtain a drawing of G' , but instead of using the bisection of the lens region, we draw every edge with angles $45^\circ + \varepsilon/2$ and $45^\circ - \varepsilon/2$ around the source vertex in counter-clockwise order and around the target vertex in clockwise order. Whenever a vertex is removed, an incoming and an outgoing edge of it is substituted by a new edge between its neighbors, so the angles at the neighbors are compatible and the new edge can inherit the direction of the old edges.

The following Lemmas is a stronger version of Lemma 1 and Theorem 1. Since the proofs do not rely on 90° angles, they can be immediately applied to the stronger version. A formal proof of Lemma 6 can be found in the arXiv version [12].

Lemma 6. *Let $G = (V, E)$ be a 4-regular plane multigraph with a plane ε -angle Lombardi drawing Γ . Then, any lens multiplication G' of G also admits a plane ε -angle Lombardi drawing.*

Lemma 7. *Let A and B be two 4-regular plane multigraphs with plane ε -angle Lombardi drawings. Let a be an edge of A and b an edge of B . Then the composition $A + B$ obtained by connecting A and B along edges a and b admits a plane ε -angle Lombardi drawing.*

Let $G = (V, E)$ be a 4-regular plane multigraph and let $x \in V$ with edges (x, a) , (x, b) , (x, c) , and (x, d) in counter-clockwise order. A *lens extension* of G is a 4-regular plane multigraph that is obtained by removing x and its incident edges from G , and adding two vertices u and v to G with two edges between u and v and the edges (u, a) , (u, b) , (v, c) , (v, d) . Informally, that means that a vertex is substituted by a lens.

Lemma 8. *Let $G = (V, E)$ be a 4-regular plane multigraph with a plane ε -angle Lombardi drawing Γ . Then, any lens extension of G admits a plane $(\varepsilon + \varepsilon')$ -angle Lombardi drawing for every $\varepsilon' > 0$.*

Sketch of Proof. Let $x \in V$ be the vertex that we want to perform the lens extension on, such that we get the edges $(u, a), (u, b), (v, c), (v, d)$ in the obtained graph G' . Let α be the angle between the tangents of (x, a) and (x, b) at x in Γ . Since Γ is a plane ε -angle Lombardi drawing, we have that $\alpha \leq 90^\circ + \varepsilon$. Further, the angle between the tangents of (x, c) and (x, d) at x in Γ is also α , while the angles between the tangents of (x, b) and (x, c) at x and between the tangents of (x, d) and (x, a) at x are both $180^\circ - \alpha$. We apply the Möbius-transformation on Γ that maps the edges (x, a) and (x, d) to straight-line segments and a lies on the same y -coordinate and to the right of x ; hence, d lies strictly below x .

We aim to place v such that the angle between the arcs (v, c) and (v, d) is $\alpha + \lambda$ for some $0 < \lambda \leq \varepsilon'$, which we will show how to choose later. We have fixed ports at c and d and a fixed angle $\alpha + \lambda$ at v . According to Property 1, all possible positions of v lie on a circle through c and d . Note that the circle through c, d , and x describes all possible positions of neighbors of c and d with angle α . Since the desired angle gets larger, the position circle for v contains a point v_d on the straight-line edge (x, d) and a point v_c on the half-line starting from x with the angle of the port used by the arc (x, c) ; see Fig. 11. We denote by C_v^λ the circular arc between v_c and v_d on the placement circle of v that gives the angle $\alpha + \lambda$ at v . We use the same construction for u to obtain the circular arc C_u^λ between u_a and u_b . Since the drawing of G is plane, there exists some non-empty region in which we can move x , such that the arcs $(x, a), (x, b), (x, c), (x, d)$ are drawn with the same ports at a, b, c, d and do not cross any other edge of the drawing. We choose λ as the largest value with $0 < \lambda \leq \varepsilon'$ such that the two circular arcs C_u and C_v lie completely inside this region.

We show how to find a pair of points on C_v^λ and C_u^λ such that we can connect them via a lens in the arXiv version [12].

Lemma 9. *Every 4-regular plane multigraph with at most 3 vertices admits a plane ε -regular Lombardi drawing for every $0 \leq \varepsilon < 90^\circ$.*

Proof. There are only two 4-regular multigraphs with at most 3 vertices and each of them has a plane Lombardi drawing as depicted in Fig. 12a. For some $0^\circ < \varepsilon < 90^\circ$, we can obtain a plane ε -regular Lombardi drawing by simply making the circular arcs larger or smaller, as depicted in Fig. 12b.

We are now ready to present the main result of this section. The proof boils down to a large case distinction using the tools developed in the previous discussion. We split the original graph into biconnected components and then use Lemma 9 and 5 as base cases. With the help of lens extensions, lens multiplications, and knot sums we can combine the “near-Lombardi” drawings of the biconnected graphs to generate an “near-Lombardi” drawing of the original graph. As a consequence, every knot is near-Lombardi. The full proof is given in the arXiv version [12].

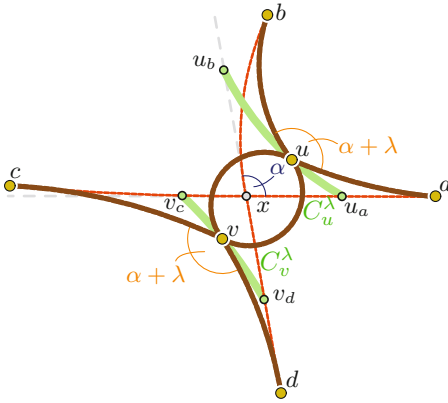


Fig. 11. The circular arc C_u^λ between u_a and u_b on the placement circles of u and the circular arc C_v^λ between v_c and v_d on the placement circles of v .

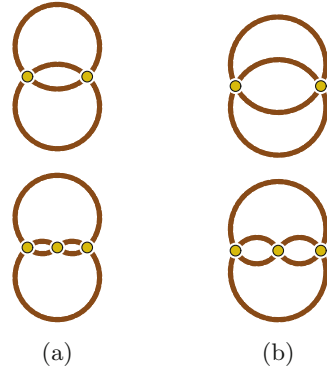


Fig. 12. The only biconnected 4-regular multigraphs with at most 3 vertices. (a) plane Lombardi and (b) plane ε -angle Lombardi drawings.

Theorem 7. *Let $G = (V, E)$ be a biconnected 4-regular plane multigraph and let $\varepsilon > 0$. Then G admits a plane ε -angle Lombardi drawing.*

Acknowledgements. Research for this work was initiated at Dagstuhl Seminar 17072 *Applications of Topology to the Analysis of 1-Dimensional Objects* which took place in February 2017. We thank Benjamin Burton for bringing the problem to our attention and Dylan Thurston for helpful discussion.

References

1. Knot drawing in Knot Atlas. http://katlas.org/wiki/Printable_Manual#Drawing_Planar_Diagrams. Accessed 08 July 2017
2. Alam, M.J., Bekos, M.A., Kaufmann, M., Kindermann, P., Kobourov, S.G., Wolff, A.: Smooth orthogonal drawings of planar graphs. In: Pardo, A., Viola, A. (eds.) LATIN 2014. LNCS, vol. 8392, pp. 144–155. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54423-1_13
3. Alexander, J.W., Briggs, G.B.: On types of knotted curves. *Ann. Math.* **28**, 562–586 (1927)
4. Bekos, M.A., Kaufmann, M., Kobourov, S.G., Symvonis, A.: Smooth orthogonal layouts. In: Didimo, W., Patrignani, M. (eds.) GD 2012. LNCS, vol. 7704, pp. 150–161. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36763-2_14
5. Cromwell, P.R.: Arc presentations of knots and links. *Banach Center Publ.* **42**, 57–64 (1998)
6. Cromwell, P.R.: *Knots and Links*. Cambridge University Press, Cambridge (2004)
7. Duncan, C.A., Eppstein, D., Goodrich, M.T., Kobourov, S.G., Löffler, M.: Planar and Poly-arc Lombardi drawings. In: van Kreveld, M., Speckmann, B. (eds.) GD 2011. LNCS, vol. 7034, pp. 308–319. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-25878-7_30

8. Duncan, C.A., Eppstein, D., Goodrich, M.T., Kobourov, S.G., Nöllenburg, M.: Lombardi drawings of graphs. *J. Graph Algorithms Appl.* **16**(1), 37–83 (2012)
9. Eppstein, D.: Planar Lombardi drawings for subcubic graphs. In: Didimo, W., Patrignani, M. (eds.) GD 2012. LNCS, vol. 7704, pp. 126–137. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36763-2_12
10. Eppstein, D.: A Möbius-invariant power diagram and its applications to soap bubbles and planar Lombardi drawing. *Discrete Comput. Geom.* **52**, 515–550 (2014)
11. Kauffman, L.H.: On knots. In: *Annals of Mathematical Studies*, vol. 115. Princeton University Press, Princeton (1987)
12. Kindermann, P., Kobourov, S., Löffler, M., Nöllenburg, M., Schulz, A., Vogtenhuber, B.: Lombardi drawings of knots and links. Arxiv report 1708.09819 (2017)
13. Liu, Y., Morgana, A., Simeone, B.: A linear algorithm for 2-bend embeddings of planar graphs in the two-dimensional grid. *Discrete Appl. Math.* **81**(1–3), 69–91 (1998)
14. Livingston, C.: Knot theory. In: *The Carus Mathematical Monographs*, vol. 24. Mathematical Association of America, Washington, DC (1993)
15. Rolfsen, D.: *Knots and Links*. American Mathematical Society, Providence (1976)
16. Scharein, R.G.: Interactive topological drawing. Ph.D. thesis, Department of Computer Science, The University of British Columbia (1998)
17. Schubert, H.: Sitzungsbericht. *Heidelberger Akad. Wiss., Math.-Naturwiss. Klasse*, 3. Abhandlung (1949)
18. Tutte, W.T.: How to draw a graph. *Proc. Lond. Math. Soc.* **13**(3), 743–768 (1963)

Arrangements of Pseudocircles: Triangles and Drawings

Stefan Felsner¹(✉) and Manfred Scheucher²

¹ Technische Universität Berlin, Berlin, Germany
felsner@math.tu-berlin.de

² Graz University of Technology, Graz, Austria
mscheuch@ist.tugraz.at

Abstract. A pseudocircle is a simple closed curve on the sphere or in the plane. The study of arrangements of pseudocircles was initiated by Grünbaum, who defined them as collections of simple closed curves that pairwise intersect in exactly two crossings. Grünbaum conjectured that the number of triangular cells p_3 in digon-free arrangements of n pairwise intersecting pseudocircles is at least $2n - 4$. We present examples to disprove this conjecture. With a recursive construction based on an example with 12 pseudocircles and 16 triangles we obtain a family with $p_3(\mathcal{A})/n \rightarrow 16/11 = 1.45$. We expect that the lower bound $p_3(\mathcal{A}) \geq 4n/3$ is tight for infinitely many simple arrangements. It may however be that digon-free arrangements of n pairwise intersecting circles indeed have at least $2n - 4$ triangles.

For pairwise intersecting arrangements with digons we have a lower bound of $p_3 \geq 2n/3$, and conjecture that $p_3 \geq n - 1$.

Concerning the maximum number of triangles in pairwise intersecting arrangements of pseudocircles, we show that $p_3 \leq 2n^2/3 + O(n)$. This is essentially best possible because families of pairwise intersecting arrangements of n pseudocircles with $p_3/n^2 \rightarrow 2/3$ as $n \rightarrow \infty$ are known.

The paper contains many drawings of arrangements of pseudocircles and a good fraction of these drawings was produced automatically from the combinatorial data produced by the generation algorithm. In the final section we describe some aspects of the drawing algorithm.

1 Introduction

Arrangements of pseudocircles generalize arrangements of circles in the same vein as arrangements of pseudolines generalize arrangements of lines. The study of arrangements of pseudolines was initiated 1918 with an article of Levi [7] where he studied triangles in arrangements. Since then arrangements of pseudolines were intensively studied and the handbook article on the topic [2] lists more than 100 references.

S. Felsner—Partially supported by DFG Grant FE 340/11-1.

M. Scheucher—Partially supported by ERC Advanced Research Grant no 267165 (DISCONV).

Grünbaum [6] initiated the study of arrangements of pseudocircles. By stating a large number of conjectures he was hoping to attract the attention of researchers for the topic. The success of this program was limited and several of Grünbaum's 45 year old conjectures remain unsettled. In this paper we report on some progress regarding conjectures involving numbers of triangles and digons in arrangements of pseudocircles.

Some of our results and new conjectures are based on a program written by the second author that enumerates all arrangements of up to 7 pairwise intersecting pseudocircles. Before formally stating our main results we introduce some terminology:

An *arrangement of pseudocircles* is a collection of closed curves in the plane or on the sphere, called *pseudocircles*, with the property that the intersection of any two of the pseudocircles is either empty or consists of two points where the curves cross. An arrangement \mathcal{A} of pseudocircles is

simple, if no three pseudocircles of \mathcal{A} intersect in a common point.

pairwise intersecting, if any two pseudocircles of \mathcal{A} have non-empty intersection. We will frequently abbreviate and just write “*intersecting*” instead of “pairwise intersecting”.

cylindrical, if there are two cells of the arrangement which are separated by each of the pseudocircles.

digon-free, if there is no cell of the arrangement which is incident to only two pseudocircles.

We consider the sphere to be the most natural ambient space for arrangements of pseudocircles. Consequently, we call two arrangements isomorphic if they induce homeomorphic cell decompositions of the sphere. In many cases, in particular in all our figures, arrangements of pseudocircles are embedded in the Euclidean plane, i.e., there is a distinguished outer/unbounded cell. An advantage of such a representation is that we can refer to the inner and outer side of a pseudocircle. Note that for every cylindrical arrangement of pseudocircles it is possible to choose the unbounded cell such that there is a point in the intersection of the interior pseudodiscs of all pseudocircles.

In an arrangement \mathcal{A} of pseudocircles, we denote a cell with k crossings on its boundary as a k -cell and let $p_k(\mathcal{A})$ be the number of k -cells of \mathcal{A} . Following Grünbaum we call 2-cells *digons* and remark that some other authors call them *lenses*. 3-cells are *triangles*, 4-cells are *quadrangles*, and 5-cells are *pentagons*.

Conjecture 3.7 from Grünbaum's monograph [6] is: *Every (not necessarily simple) digon-free arrangement of n pairwise intersecting pseudocircles has at least $2n - 4$ triangles.* Grünbaum also provides examples of arrangements with $n \geq 6$ pseudocircles and $2n - 4$ triangles.

Snoeyink and Hershberger [10] showed that the sweeping technique, which serves as an important tool for the study of arrangements of lines and pseudolines, can be adapted to work also in the case of arrangements of pseudocircles. They used sweeps to show that, in an intersecting arrangement of pseudocircles, every pseudocircle is incident to two cells which are digons or triangles on either side.

Therefore, $2p_2 + 3p_3 \geq 4n$, and whence, every intersecting digon-free arrangement of n pseudocircles has at least $4n/3$ triangles.

Felsner and Kriegel [3] observed that the bound from [10] also applies to non-simple intersecting digon-free arrangements and gave examples of arrangements showing that the bound is tight on this class for infinitely many values of n . These examples disprove the conjecture in the non-simple case.

In Sect. 2, we give counterexamples to Grünbaum’s conjecture which are simple. With a recursive construction based on an example with 12 pseudocircles and 16 triangles we obtain a family with $p_3/n \xrightarrow{n \rightarrow \infty} 16/11 = 1.45$. We then replace Grünbaum’s conjecture by Conjecture 2: *The lower bound $p_3(\mathcal{A}) \geq 4n/3$ is tight for infinitely many non-isomorphic simple arrangements.*

A specific arrangement \mathcal{N}_6 of 6 pseudocircles with 8 triangles appears as a subarrangement in all known simple, intersecting, digon-free arrangements with $p_3 < 2n - 4$. From [5] it is known that \mathcal{N}_6 is not circularizable, i.e., not representable by circles. This motivates the question, whether indeed Grünbaum’s conjecture is true when restricted to intersecting arrangements of circles, see Conjecture 1. In Subsect. 2.1 we discuss arrangements with digons. We give an easy extension of the argument of Snoeyink and Hershberger [10] to show that these arrangements contain at least $2n/3$ triangles. All arrangements known to us have at least $n - 1$ triangles and therefore our Conjecture 3 is that $n - 1$ is a tight lower bound for intersecting arrangements with digons.

In Sect. 3 we study the maximum number of triangles in arrangements of n pseudocircles. We show an upper bound of order $2n^2/3 + O(n)$. For the lower bound construction we glue two arrangements of n pseudolines into an arrangement of n pseudocircles. Since respective arrangements of pseudolines are known, we obtain arrangements of pseudocircles with $2n(n - 1)/3$ triangles for $n \equiv 0, 4 \pmod{6}$.

The paper contains many drawings of arrangements of pseudocircles and a good fraction of these drawings was produced automatically from the combinatorial data produced by the generation algorithm. In Sect. 4 we describe some aspects of the drawing algorithm which is based on iterative calls to a Tutte embedding a.k.a. spring embedding with adapting weights on the edges.

From now on (unless explicitly stated otherwise) the term *arrangement* is used as equivalent to *simple arrangement of pairwise intersecting pseudocircles*.

2 Arrangements with Few Triangles

The main result of this section is the following theorem, which disproves Grünbaum’s conjecture.

Theorem 1. *The minimum number of triangles in digon-free arrangements of n pseudocircles is*

- (i) 8 for $3 \leq n \leq 6$.
- (ii) $\lceil \frac{4}{3}n \rceil$ for $6 \leq n \leq 14$.
- (iii) $< \frac{16}{11}n$ for all $n = 11k + 1$ with $k \in \mathbb{N}$.

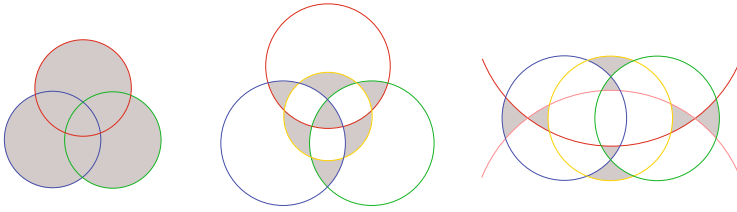


Fig. 1. Arrangements of $n = 3, 4, 5$ circles and $p_3 = 8$ triangles each. Triangles (except the outer face) are colored gray.

Figures 1 and 2 show arrangements with the minimum number of triangles for up to 8 pseudocircles. We remark that, in total, there are three non-isomorphic arrangements of $n = 8$ pseudocircles with $p_3 = 11$ triangles, these are the smallest counterexamples to Grünbaum’s conjecture (cf. Lemma 1). We refer to our website [8] for further examples.

The basis for Theorem 1 was laid by exhaustive computations, which generated all simple arrangements of up to $n = 7$ pseudocircles. Starting with the unique arrangement of two intersecting pseudocircles, our program recursively inserted pseudocircles in all possible ways. Since counting arrangements is also interesting, we state the numbers in Table 1. The table shows the number of simple intersecting pseudocircle arrangements on the sphere. The first row shows the numbers when digons are allowed and the second row shows the numbers of digon-free arrangements. The arrangements and more information can be found on the companion website [8].

Table 1. Number of combinatorially different arrangements of n pseudocircles.

n	2	3	4	5	6	7
General	1	2	8	278	145 058	447 905 202
Digon-free	0	1	2	14	2 131	3 012 972

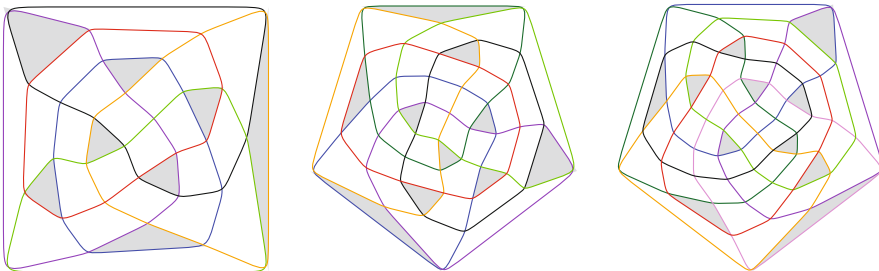


Fig. 2. Arrangements with $n = 6, 7, 8$ and $8, 10, 11$ triangles respectively.

From the complete enumeration we know the minimum number of triangles for $n \leq 7$. In the range from 8 to 14, we iteratively used arrangements with n pseudocircles and a small number of triangles and digons to generate arrangements with $n + 1$ pseudocircles and the same property. Using this strategy, we found arrangements with $\lceil 4n/3 \rceil$ triangles for all n in this range. The corresponding lower bound $p_3(\mathcal{A}) \geq 4n/3$ is known from [10].

A result of the computations was that the triangle-minimizing example for $n = 6$ is unique, i.e., there is a unique simple arrangement \mathcal{N}_6 with 6 pseudocircles and only 8 triangles. In [5] we have shown that \mathcal{N}_6 is not circularizable. The arrangement \mathcal{N}_6 is a subarrangement of all known arrangements with less than $2n - 4$ triangles. Therefore, the following weakening of Grünbaum’s conjecture may be true.

Conjecture 1 (Weak Grünbaum Conjecture). Every digon-free arrangement of n circles has at least $2n - 4$ triangles.

We know that this conjecture is true for all $n \leq 9$. The claim, that we have checked all arrangements with $p_3(\mathcal{A}) < 2n - 4$ in this range, is justified by the following lemma, which restricts the pairs (p_2, p_3) for which there exist arrangements of n pseudocircles whose extensions have $p_3(\mathcal{A}) < 2n - 4$. In particular, to get all digon-free arrangements with $n = 9$ and 12 triangles we only had to extend arrangements with $n = 7$ and $n = 8$, where $p_3 + 2p_2 \leq 12$. It turned out, that all arrangements on $n = 9$ pseudocircles with 12 triangles are non-circularizable since all of them contain \mathcal{N}_6 as a subarrangement.

Lemma 1. *Let \mathcal{A} be an arrangement of pseudocircles. Then for every subarrangement \mathcal{A}' of \mathcal{A} we have*

$$p_3(\mathcal{A}') + 2p_2(\mathcal{A}') \leq p_3(\mathcal{A}) + 2p_2(\mathcal{A}).$$

Proof. We show the statement for a subarrangement \mathcal{A}' in which one pseudocircle C is removed from \mathcal{A} . The inequality then follows by iterating the argument. The arrangement \mathcal{A}' partitions the pseudocircle C into arcs. Reinsert these arcs one by one.

Consider a triangle of \mathcal{A}' . After adding an arc, one of the following cases occurs: (1) the triangle remains untouched, or (2) the triangle is split into a triangle and a quadrangle, or (3) a digon is created in the region of the triangle.

Now consider a digon of \mathcal{A}' . After adding an arc, either (1) there is a new digon inside this digon, or (2) the digon has been split into two triangles. \square

We now prepare for the proof of Theorem 1 (iii). The basis of the construction is the arrangement \mathcal{A}_{12} with 12 pseudocircles and 16 triangles shown in Fig. 3a. This arrangement will be used iteratively for a ‘merge’ as described by the following lemma.

Lemma 2. *Let \mathcal{A} and \mathcal{B} be digon-free arrangements of $n_{\mathcal{A}} \geq 3$ and $n_{\mathcal{B}} \geq 3$ pseudocircles, respectively. If there is a simple curve $P_{\mathcal{A}}$ that (1) intersects every pseudocircle of \mathcal{A} exactly once (2) contains no vertex of \mathcal{A} , (3) traverses $\tau \geq 1$*

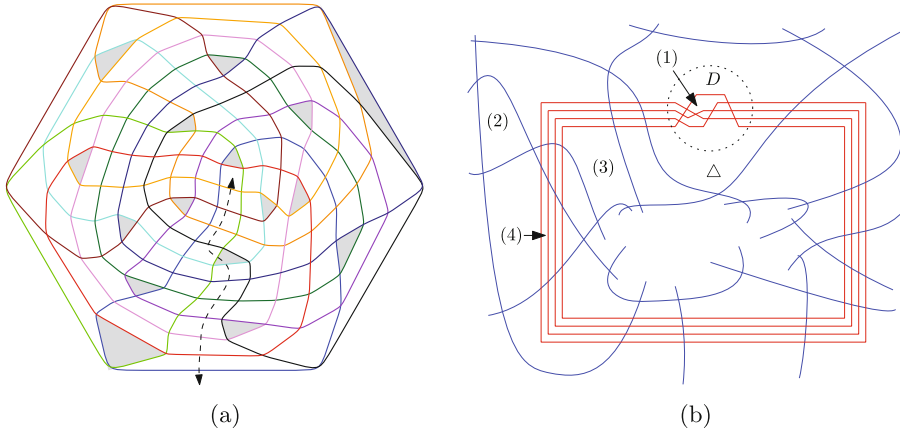


Fig. 3. (a) A digon-free, intersecting arrangement \mathcal{A}_{12} of $n = 12$ pseudocircles with exactly 16 triangles. The dashed curve intersects every pseudocircle exactly once. (b) An illustration of the construction in Lemma 2. Pseudocircles of \mathcal{A} (\mathcal{B}) are drawn red (blue). (Colour figure online)

triangles of \mathcal{A} , and (4) forms δ triangles with pairs of pseudocircles from \mathcal{A} , then there is a digon-free arrangement \mathcal{C} of $n_{\mathcal{A}} + n_{\mathcal{B}} - 1$ pseudocircles with $p_3(\mathcal{C}) = p_3(\mathcal{A}) + p_3(\mathcal{B}) + \delta - \tau - 1$ triangles.

Proof. Take a drawing of \mathcal{A} and make a hole in the two cells, which contain the ends of $P_{\mathcal{A}}$. This yields a drawing of \mathcal{A} on a cylinder such that none of the pseudocircles is contractible. The path $P_{\mathcal{A}}$ connects the two boundaries of the cylinder. In fact, the existence of a path with the properties of $P_{\mathcal{A}}$ is characterizing cylindrical arrangements.

Stretch the cylindrical drawing such that it becomes a narrow belt, where all intersections of pseudocircles take place in a small disk, which we call *belt-buckle*. This drawing of \mathcal{A} is called a *belt drawing*. The drawing of the red subarrangement in Fig. 3b shows a belt drawing.

Choose a triangle Δ in \mathcal{B} and a pseudocircle B which is incident to Δ . Let b be the *edge* of B on the boundary of Δ . Specify a disk D , which is traversed by b and disjoint from all other edges of \mathcal{B} . Now replace B by a belt drawing of \mathcal{A} in a small neighborhood of B such that the belt-buckle is drawn within D ; see Fig. 3b.

The arrangement \mathcal{C} obtained from *merging* \mathcal{A} and \mathcal{B} , as we just described, has $n_{\mathcal{A}} + n_{\mathcal{B}} - 1$ pseudocircles. Moreover if \mathcal{A} and \mathcal{B} are digon-free/intersecting, then \mathcal{C} has the same property. Most of the cells c of \mathcal{C} are of one of the following four types:

- (a) All boundary edges of c belong to pseudocircles of \mathcal{A} .
- (b) All boundary edges of c belong to pseudocircles of \mathcal{B} .

- (c) All but one of the boundary edges of c belong to pseudocircles of \mathcal{B} and the remaining edge belongs to \mathcal{A} . (These cells correspond to cells of \mathcal{B} with a boundary edge on B .)
- (d) Quadrangular cells, whose boundary edges alternatingly belong to \mathcal{A} and \mathcal{B} .

From the cells of \mathcal{B} , only \triangle and the other cell containing b (which is not a triangle since \mathcal{B} is digon-free) have not been taken into account. In \mathcal{C} , the corresponding two cells have at least two boundary edges from \mathcal{B} and at least two from \mathcal{A} . Consequently, neither of the two cells are triangles. The remaining cells of \mathcal{C} are bounded by pseudocircles from \mathcal{A} together with one of the two bounding pseudocircles of \triangle other than B . These two pseudocircles cross through \mathcal{A} following the path prescribed by $P_{\mathcal{A}}$. There are δ triangles among these cells, but τ of these are obtained because $P_{\mathcal{A}}$ traverses a triangle of \mathcal{A} . Among cells of \mathcal{C} of types (1) to (4) all the triangles have a corresponding triangle in \mathcal{A} or \mathcal{B} . But \triangle is a triangle of \mathcal{B} which does not occur in this correspondence. Hence, there are $p_3(\mathcal{A}) + p_3(\mathcal{B}) + \delta - \tau - 1$ triangles in \mathcal{C} . \square

Proof of Theorem 1 (iii). We use \mathcal{A}_{12} , the arrangement shown in Fig. 3a, in the role of \mathcal{A} for our recursive construction. The dashed path in the figure is used as $P_{\mathcal{A}}$ with $\delta = 2$ and $\tau = 1$. Starting with $\mathcal{C}_1 = \mathcal{A}_{12}$ and defining \mathcal{C}_{k+1} as the merge of \mathcal{C}_k and \mathcal{A}_{12} , we construct a sequence $\{\mathcal{C}_k\}_{k \in \mathbb{N}}$ of digon-free arrangements with $n(\mathcal{C}_k) = 11k + 1$ pseudocircles and $p_3(\mathcal{C}_k) = 16k$ triangles. The fraction $16k/(11k + 1)$ is increasing with k and converges to $16/11 = 1.\overline{45}$ as n goes to ∞ . \square

We remark that using other arrangements from Theorem 1 (ii) (which also admit a path with $\delta = 2$ and $\tau = 1$) in the recursion, we obtain arrangements with $p_3 = \lceil \frac{16}{11}n \rceil$ triangles for all $n \geq 6$.

Since the lower bound $\lceil \frac{4}{3}n \rceil$ is tight for $6 \leq n \leq 14$, we believe that the following is true:

Conjecture 2. There are digon-free arrangements \mathcal{A} with $p_3(\mathcal{A}) = \lceil 4n/3 \rceil$ for infinitely many values of n .

2.1 Arrangements with Digons

We know arrangements of n pseudocircles with digons and only $n - 1$ triangles. The example shown in Fig. 4a is part of an infinite family of such arrangements.

Using ideas based on sweeps (cf. [10]), we can show that every pseudocircle is incident to at least two triangles. This implies the following theorem:

Theorem 2. *Every arrangement of $n \geq 3$ pseudocircles has at least $2n/3$ triangles.*

The proof of the theorem is based on the following lemma:

Lemma 3. *Let C be a pseudocircle in an arrangement of $n \geq 3$ pseudocircles. Then all digons incident to C lie on the same side of C .*

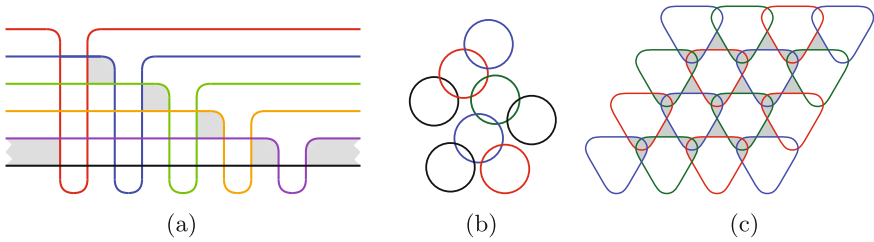


Fig. 4. Example arrangements (a) n pseudocircles with n digons and $n - 1$ triangles (b) “trees of circles” with no triangles (c) connected arrangements of n pseudocircles with triangle-cell-ratio of $\frac{5}{6} - O(\frac{1}{\sqrt{n}})$.

Proof. Consider a pseudocircle C' that forms a digon D' with C that lies, say, “inside” C . If C'' also forms a digon D'' , then C'' has to cross C' in the exterior of C . Hence D'' also has to lie “inside” C . Consequently, all digons incident to C lie on the same side of C . \square

Proof of Theorem 2. Let \mathcal{A} be an arrangement and consider a drawing of \mathcal{A} in the plane. Snoeyink and Hershberger [10] have shown that starting with any circle C from \mathcal{A} the outside of C can be swept with a closed curve γ until all of the arrangement is inside of γ . During the sweep γ is intersecting every pseudocircle from \mathcal{A} at most twice. The sweep uses two types¹ of move to make progress:

- (1) *take a crossing*, in [10] this is called ‘pass a triangle’;
- (2) *leave a pseudocircle*, this is possible when γ and some pseudocircle form a digon which is on the outside of γ , in [10] this is called ‘pass a hump’.

Let C be a pseudocircle of \mathcal{A} . By the previous lemma, all digons incident to C lie on the same side of C . Redraw \mathcal{A} so that all digons incident to C are inside C . The first move of a sweep starting at C has to take a crossing, and hence, there is a triangle Δ incident to C . Redraw \mathcal{A} such that Δ becomes the unbounded face. Again consider a sweep starting at C . The first move of this sweep reveals a triangle Δ' incident to C . Since Δ is not a bounded triangle of the new drawing we have $\Delta \neq \Delta'$, and hence, C is incident to at least two triangles. The proof is completed by double counting the number of incidences of triangles and pseudocircles. \square

Since for $3 \leq n \leq 7$ every arrangement has at least $n - 1$ triangles, we believe that the following is true:

Conjecture 3. Every intersecting arrangement of $n \geq 3$ pseudocircles has at least $n - 1$ triangles.

¹ There is a third type of move for sweeps of arrangements of pseudocircles, it is called *take a hump* and does not occur in our case, as each two pseudocircles already intersect.

If the arrangement is not required to be intersecting, then the proof of Lemma 3 fails and indeed there are examples of non-intersecting arrangements without triangles, e.g., a “tree of circles”, see Fig. 4b.

3 Maximum Number of Triangles

Regarding the maximal number of triangles the complete enumeration provides precise data for $n \leq 7$. We used heuristics to generate examples with many triangles for larger n . Table 2 and Fig. 5 shows the results. For $n \geq 4$ there is only one instance where we know an arrangement with more than $\frac{4}{3}\binom{n}{2}$ triangles. This number is $1/3$ times the number of edges of the arrangement, i.e., it is an upper bound for the number of triangles in arrangements where each edge is incident to at most one triangle. In the next subsection we show that asymptotically the contribution of edges that are incident to two triangles is neglectable. The last subsection gives a construction of arrangements which show that $\lfloor \frac{4}{3}\binom{n}{2} \rfloor$ is attained for infinitely many values of n .

Table 2. Upper bound on the number of triangles.

	2	3	4	5	6	7	8	9	10
Simple	0	8	8	13	20	29	≥ 37	≥ 48	≥ 60
Digon-free	–	8	8	12	20	29	≥ 37	≥ 48	≥ 60
$\lfloor \frac{4}{3}\binom{n}{2} \rfloor$	1	4	8	13	20	28	37	48	60

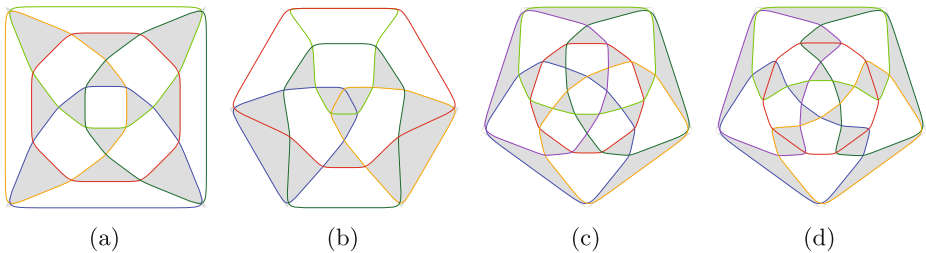


Fig. 5. (a) and (b) show arrangements with $n = 5$ pseudocircles. The first one is digon-free and has 12 triangles and the second one has 13 triangles and one digon. (c) and (d) show arrangements with $n = 6$ and 20 triangles. The arrangement in (c) is the skeleton of the Icosidodecahedron.

Recall that we only study simple arrangements. Grünbaum [6] also looked at non-simple arrangements. His Figures 3.30, 3.31, and 3.32 show drawings of simplicial arrangements that have $n = 7$ with $p_3 = 32$, $n = 8$ with $p_3 = 50$, and $n = 9$ with $p_3 = 62$, respectively. Hence, non-simple arrangements can have more triangles.

Theorem 3. $p_3(\mathcal{A}) \leq \frac{2}{3}n^2 + O(n)$.

The proof of this theorem can be found in the appendix of the version submitted to the arXiv [4].

Remarks

- Since intersecting arrangements have $2\binom{n}{2} + 2 = n^2 - O(n)$ faces we can also state the bound as: at most $\frac{2}{3} + O(\frac{1}{n})$ of all cells of an arrangement are triangles. However, this is not true if we consider non-intersecting arrangements. Figure 4c shows a construction where this ratio converges to $\frac{5}{6}$ as $n \rightarrow \infty$. It can be shown with a counting argument that $\frac{5}{6}$ is an upper bound for the triangle-cell-ratio of simple arrangements.
- It would be interesting to get more precise results. In particular, we would like to know whether $p_3 \leq \frac{4}{3}\binom{n}{2} + O(1)$ is true for all n .

3.1 Constructions Using Arrangements of Pseudolines

Great circles on the sphere are a well known model for projective arrangements of lines. Antipodal pairs of points on the sphere correspond to points of the projective plane. Hence, the great circle arrangement corresponding to a projective arrangement \mathcal{A} of lines has twice as many vertices, edges, and faces of every type as \mathcal{A} . The same idea can be applied to projective arrangement of pseudolines. If \mathcal{A} is a projective arrangement of pseudolines take a drawing of \mathcal{A} in the unit disk D such that every line ℓ of \mathcal{A} connects two antipodal points of D . Project D to the upper hemisphere of a sphere S , such that the boundary of D becomes the equator of S . Use a projection through the center of ℓ to copy the drawing from the upper hemisphere to the lower hemisphere of S . By construction the two copies of a pseudoline ℓ from \mathcal{A} join together to form a pseudocircle. The collection of these pseudocircles yields an arrangement of pseudocircles on the sphere with twice as many vertices, edges, and faces of every type as \mathcal{A} . Arrangements of pseudocircles obtained by this construction have a special property:

- If three pseudocircles C , C' , and C'' have no common crossing, then C'' separates the two crossings of C and C' .

Grünbaum calls arrangements with this property ‘symmetric’. In the context of oriented matroids the property is part of the definition of arrangements of pseudocircles.

Arrangements of pseudolines which maximize the number of triangles have been studied intensively. The end of this line of research is marked by Blanc [1]. This paper gives precise bounds for the maximum both in the Euclidean and in the projective case. In particular, Blanc constructs examples of projective arrangements of pseudolines with $\frac{2}{3}\binom{n}{2}$ triangles for an infinite number of values of n . This directly yields arrangements of pseudocircles with $\frac{4}{3}\binom{n}{2}$ triangles. The ‘doubling method’ that has been used for constructions of arrangements of pseudolines with many triangles, see [1], can also be applied for pseudocircles. In fact, in the case of pseudocircles there is more flexibility for applying the method. Therefore, it is possible that $\lfloor \frac{4}{3}\binom{n}{2} \rfloor$ triangles can be achieved for all n .

4 Visualization

Most of the figures in this paper have been automatically generated by our framework, which was written in the mathematical software SageMath [11] and is available on demand. We encode an arrangement of pseudocircles by its dual graph. Each face in the arrangement is represented by a vertex and two vertices share an edge if and only if the two corresponding faces share a common pseudosegment. As our arrangements are intersecting, it is easy to see that the dual graph is 3-connected and thus its embedding is unique on the sphere (up to isomorphism).

To visualize an arrangement of pseudocircles, we draw the primal (multi)graph using straight-line segments, in which vertices represent crossings of pseudocircles and edges connect two vertices if they are connected by a pseudocircle segment. Note that in the presence of digons we obtain double-edges.

In our drawings, pseudocircles are colored by distinct colors, and triangles (except the outer face) are filled gray. In straight-line drawings, edges corresponding to digons are drawn dashed in the two respective colors alternatingly, while in the curved drawings digons are represented by a point where the two respective pseudocircles touch.

4.1 Iterated Tutte Embeddings

To generate nice aesthetic drawings automatically, we iteratively use weighted Tutte embeddings. We fix a non-digon cell as the outer cell and arrange the vertices of the outer cell as the corners of a regular polygon. Starting with edge-weights all equal to 1, we obtain an ordinary plane Tutte embedding.

For iteration j , we set the weights (force of attraction) of an edge $e = \{u, v\}$ proportional to $p(A(f_1)) + p(A(f_2)) + q(\|u - v\|/j)$ where f_1, f_2 are the faces incident to e , $A(\cdot)$ is the area function, $\|\cdot\|$ is the Euclidean norm, and p, q are suitable monotonically increasing functions from \mathbb{R}^+ to \mathbb{R}^+ (we use $p(x) = x^4$ and $q(x) = x^2/10$).

Intuitively, if the area of a face becomes too large, the weights of its incident edges are increased and will rather be shorter so that the area of the face will also get smaller in the next iteration. It turned out that in some cases the areas of the faces became well balanced but some edges were very short and others long. Therefore we added the dependence on the edge length which is strong at the beginning and decreases with the iterations. The particular choice of the functions was the result of interactive tuning. The iteration is terminated when the change of the weights is small.

4.2 Visualization Using Curves

On the basis of the straight-line embedding obtained with the Tutte iteration we use splines to smoothen the curves. The details are as follows. First we take a 2-subdivision of the graph, where all subdivision-vertices adjacent to a given vertex v are placed at the same distance $d(v)$ from v . We choose $d(v)$ so that

it is at most $1/3$ of the length of an edge incident to v . We then use B-splines to visualize the curves. Even though one can draw Bézier curves directly with Sage, we mostly generated ipe files (xml-format) so that we can further process the arrangements. Figures 6a and b show the straight-line and curved drawing of the same arrangement.

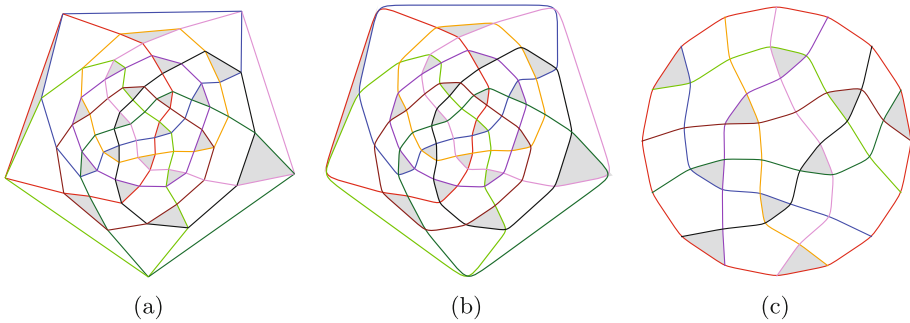


Fig. 6. (a) Straight-line and (b) curved drawings of the arrangement of pseudo (great) circles, which consists of two copies of (c) the (non-stretchable) non-Pappus pseudoline arrangement of pseudolines.

4.3 Visualization of Arrangements of Pseudolines

We also adopted the code to visualize arrangements of pseudolines nicely. One of the lines is considered as the “line at infinity” which is then drawn as a regular polygon. Figure 6c gives an illustration.

For arrangements of pseudolines we used the framework pyotlib, which originated from the Bachelor’s thesis of Scheucher [9].

References

1. Blanc, J.: The best polynomial bounds for the number of triangles in a simple arrangement of n pseudo-lines. *Geoinformatics* **21**, 5–17 (2011)
2. Felsner, S., Goodman, J.E.: Pseudoline arrangements. In: Toth, C.D., O’Rourke, J., Goodman, J.E. (eds.) *Handbook of Discrete and Computational Geometry*, 3rd edn. CRC Press, Boca Raton (2016)
3. Felsner, S., Kriegel, K.: Triangles in Euclidean arrangements. In: Hromkovič, J., Sýkora, O. (eds.) *WG 1998*. LNCS, vol. 1517, pp. 137–148. Springer, Heidelberg (1998). https://doi.org/10.1007/10692760_12
4. Felsner, S., Scheucher, M.: Arrangements of pseudocircles: triangles and drawings. <http://arXiv.org/abs/1708.06449v2> (2017)
5. Felsner, S., Scheucher, M.: Triangles in arrangements of pseudocircles. In: *Proceedings of 33rd European Workshop on Computational Geometry (EuroCG 2017)*, pp. 225–228 (2017). <http://csoconferences.mah.se/eurocg2017/proceedings.pdf>
6. Grünbaum, B.: *Arrangements and Spreads*. Reg. Conf. Ser. Math. AMS (1972). reprinted 1980

7. Levi, F.: Die Teilung der projektiven Ebene durch Gerade oder Pseudogerade. Ber. Math. Phys. Kl. sächs. Akad. Wiss. Leipzig **78**, 256–267 (1926)
8. Scheucher, M.: http://www.ist.tugraz.at/scheucher/arrangements_of_pseudocircles/
9. Scheucher, M.: On Order Types, Projective Classes, and Realizations, Bachelor's thesis (2014)
10. Snoeyink, J., Hershberger, J.: Sweeping arrangements of curves. In: Goodman, J.E., Pollack, R., Steiger, W. (eds.) Discrete and Computational Geometry, DIMACS series in discrete mathematics and theoretical computer science, vol. 6, pp. 309–349. AMS (1991)
11. Stein, W., et al.: Sage Mathematics Software (Version 7.6). The Sage Development Team (2017). <http://www.sagemath.org>

Drawing Bobbin Lace Graphs, or, Fundamental Cycles for a Subclass of Periodic Graphs

Therese Biedl and Veronika Irvine^(✉)

David R. Cheriton School of Computer Science, University of Waterloo,
Waterloo, Canada
{biedl,virvine}@uwaterloo.ca

Abstract. In this paper, we study a class of graph drawings that arise from bobbin lace patterns. The drawings are periodic and require a combinatorial embedding with specific properties which we outline and demonstrate can be verified in linear time. In addition, a lace graph drawing has a topological requirement: it contains a set of non-contractible directed cycles which must be homotopic to $(1, 0)$, that is, when drawn on a torus, each cycle wraps once around the minor meridian axis and zero times around the major longitude axis. We provide an algorithm for finding the two fundamental cycles of a canonical rectangular schema in a supergraph that enforces this topological constraint. The polygonal schema is then used to produce a straight-line drawing of the lace graph inside a rectangular frame. We argue that such a polygonal schema always exists for combinatorial embeddings satisfying the conditions of bobbin lace patterns, and that we can therefore create a pattern, given a graph with a fixed combinatorial embedding of genus one.

1 Introduction

Bobbin lace is a 500-year-old fibre art-form created by braiding threads together in complex patterns. See Fig. 1. Bobbin lace can depict landscapes, figures, flowers, as well geometric and abstract designs. Common to all bobbin lace compositions is the use of doubly periodic patterns to fill regions of any shape or size. It is the study of these periodic patterns that we pursue here. To create bobbin lace, threads, which are wound around wooden bobbins to facilitate handling, are arranged left to right in linear order $t_1, t_2, \dots, t_{2n-1}, t_{2n}$. The lacemaker selects four consecutive threads, starting at an odd index, and crosses the four threads over and under each other to form an alternating braid. After one or several crossings are made in this manner, the four threads are set aside and another set of four (not the same four, but possibly using a subset of the original four) is selected, again starting at an odd index, and braided. Since the selected threads are consecutive starting at an odd index, we can describe the pattern by tracking the movement of pairs of threads rather than individual strands.

T. Biedl and V. Irvine—Research supported by NSERC. Thank you to Anna Lubiw for helpful input.

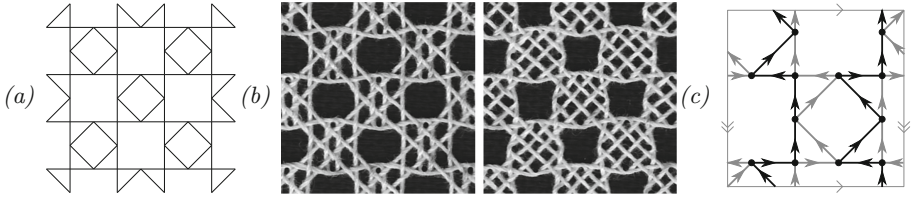


Fig. 1. Bobbin lace. (a) Pattern, (b) Lace with two different $\zeta(v)$ mappings, (c) Periodic graph drawing. Four osculating circuits distinguished using black and gray.

The application of mathematics to the study of fibre arts dates back to the beginnings of computer science. A survey of various areas, including knitting and weaving, is presented by Belcastro and Yackel [3]. Grishanov et al. take a deep look into knot theory and its relevance to the topology of textiles [6]. The second author and Ruskey [7] were the first to develop a mathematical model for bobbin lace and express its patterns using graph drawings. Specifically, a lace pattern can be represented as $(\Gamma(G), \zeta(v))$ where G is a combinatorial embedding that captures the flow of pairs of threads from one grouping of four to another, $\Gamma(G)$ gives a specific drawing of G which assigns a geometry to the position of the braids, and $\zeta(v)$ is a mapping from each node $v \in V(G)$ to a mathematical braid word which specifies the over and under crossings performed on the subset of four threads that meet at v . A systematic exploration of different $\zeta(v)$ mappings is straight-forward, although time consuming, and has been undertaken by lacemakers for several traditional patterns; see [7] for more details. Discovering which pairs of threads can be successfully combined, as represented by $\Gamma(G)$, is a much harder task and is therefore the focus of this paper.

The main question investigated in this paper is to decide, for a directed graph with a fixed rotation system, whether we can construct a straight-line drawing that is a lace pattern. We argue that recognizing such graphs can be done in linear time and depends only on their combinatorial structure (some of these results were reported earlier [7]). Lace patterns are doubly periodic. As a result, a lace graph can be drawn on the surface of a torus and lifted to its universal cover, an infinite, periodic, planar graph. A toroidal embedding can be drawn as a straight-line planar graph drawing with a rectangular outer face by choosing a canonical rectangular schema [1, 4]. The challenge is to find two fundamental cycles to serve as the borders of the rectangle under the constraints of the topological requirements of the lace pattern and restrictions imposed by the drawing algorithm itself. We show how to find a suitable polygonal schema for any valid combinatorial embedding.

2 Mathematical Model of Bobbin Lace

We assume familiarity with graphs and combinatorial embeddings; see for example [8]. Throughout this paper, $G = (V, E)$ is a directed graph that comes with

a rotation system, i.e., a clockwise order of edges around each vertex. We will assume that the rotation system describes a cellular combinatorial embedding on an orientable surface (i.e., the complement of G on the orientable surface is a collection of open topological disks). The genus of the cellular embedding can be determined from the rotation system by computing the *facial walk* consisting of edges and vertices incident to each face in order while walking around the face. All embeddings of interest here have an Euler characteristic of 0.

A circuit is a closed path in which a vertex may be visited more than once but no edges are repeated. A cycle is a closed path in which each vertex and edge is only visited once. A *contractible cycle* is a cycle that can be continuously retracted to a point. A canonical rectangular representation of a torus is bounded by a pair of *non-contractible* cycles, also referred to as fundamental cycles, that have only one vertex in common, their point of intersection.

2.1 Conditions on Lace Pattern Graph Embeddings

In a lace pattern graph, every vertex v must have exactly two incoming and two outgoing edges, corresponding to two pairs of threads that meet at v , are braided together and then separate. We call such a graph a 2-2-regular digraph.

Lace patterns are doubly periodic, i.e., they can tile the plane by translation in two non-parallel directions. One “tile” of the repeat can be drawn using a canonical rectangular schema in which the position of a thread intersecting the horizontal boundaries of the schema has the same abscissa top and bottom and a thread intersecting the vertical boundaries of the schema has the same ordinate value left and right. In graph drawing this is called periodic.

We do not want lace created from a pattern to “fall apart”, i.e., the graph needs to be suitably connected. In graph-theoretical terms, this means G must be a cellular embedding.

For a lace pattern to be workable, there must exist a partial ordering of the vertices such that, for any directed edge, the braid mapped to the tail vertex is worked before that of the head. The pattern, when repeated over the infinite plane, cannot contain directed cycles, or equivalently, the associated graph on the torus must be free from contractible directed circuits.

The graph may have loops but they must be non-contractible. Parallel edges with the same orientation do not represent a change in the four threads being braided and therefore do not appear in a lace graph. Parallel edges with opposite orientation must form a non-contractible directed cycle. In other words, a lace graph may be a multigraph but no loop or parallel edge can be a face.

In summary, the following three conditions are required for the combinatorial embedding of any lace pattern graph G :

- C1. G is a directed 2-2-regular digraph.
- C2. The rotation system of G describes a toroidal cellular embedding in which all facial walks contain at least 3 edges.
- C3. All directed circuits of G are non-contractible.

It is easy to check in linear time whether (C1) holds. To test (C2), we first compute the facial walks to ascertain the number of faces, and can then determine whether the embedding is toroidal since, by Euler’s formula, such an embedding with n vertices and $2n$ edges must have exactly n faces.

To test (C3) in linear time, we take advantage of the regular structure of our digraph via the following lemma:

Lemma 1. *Presume a 2-2-regular digraph has a toroidal cellular embedding G . If G has a contractible directed circuit C , then G will have at least one face bounded by a contractible directed circuit.*

Proof. (Sketch) Arbitrarily declare one face to contain the origin so that “inside” and “outside” are well-defined for any contractible directed circuit C . Let O_C and I_C be the number of faces outside and inside C , and consider a directed contractible circuit C that maximizes $|O_C - I_C|$. Prove, by contradiction, that a directed cycle containing an edge in its interior cannot maximize $|O_C - I_C|$ and therefore the inside of C is a face as required. Details are in the full version. □

It follows that to verify condition (C3), we simply test whether any facial walk is a directed circuit. Clearly this takes linear time.

There is one more important restriction for a workable bobbin lace pattern. The threads in the periodic pattern are continuous; threads are neither removed (by cutting) nor added (by knotting or weaving in). In other words, to fill a rectangle of fixed width and undetermined height, a fixed set of threads starts at the top of the rectangle and the same set of threads terminates (not necessarily in the same order) at the bottom of the rectangle. To achieve this, the threads in one repeat of the pattern must not have a net drift to the right or left. See [7] for a more detailed discussion of this *thread conservation* property.

To formulate the thread conservation property in a mathematically precise way, we require additional terminology and some observations resulting from (C1, C2, C3) which we will describe in the next section.

2.2 Osculating Circuits and Thread Conservation

Fix a digraph G with a combinatorial embedding such that (C1, C2, C3) hold. There are two ways in which edges can be arranged at a vertex v with $\text{indeg}(v) = \text{outdeg}(v) = 2$: Either *rotationally alternating* in which edges alternate between incoming and outgoing directions or *rotationally consecutive* with edges in the order incoming, incoming, outgoing, outgoing. Irvine and Ruskey [7] showed that the following condition is necessary for (C3):

C3'. At all vertices of G , the outgoing arcs are rotationally consecutive.

Under (C3'), we define the *left/right incoming/outgoing* edges of v as follows: going in clockwise order around v , we encounter first the left outgoing edge, then the right outgoing edge, then the right incoming edge and finally the left incoming edge. Consider two edge-disjoint directed circuits C_1, C_2 that have a

vertex v in common. There are two possible ways in which C_1 and C_2 can meet at v . In an *osculating intersection* these circuits only touch (“kiss”), i.e., both incoming and outgoing edges of C_1 are on the left side of v and the edges of C_2 are on the right side of v (or vice versa). In contrast, at a *transverse intersection* the circuits truly cross, i.e., C_1 enters from the left side of v and exits from the right, C_2 enters from the right and exits from the left (or vice versa).

Lemma 2. *Presume (C1,C2,C3) hold. The edges of G can be partitioned into a set $\mathcal{P}(G)$ of disjoint directed circuits such that no two circuits in $\mathcal{P}(G)$ have a transverse intersection. Furthermore, this partition is unique and can be found in linear time.*

Proof. Arbitrarily select an edge e_1 of G as the start of a circuit P_1 . If this edge is left incoming at its head v , then let e_2 be the left outgoing edge at v , else let e_2 be the right outgoing edge at v . Put differently, e_1 and e_2 are on the “same side” of v . Append e_2 to P_1 and repeat the operation at the head of e_2 . Since the graph is finite, we eventually must close the circuit P_1 ; this is the first element of the partition $\mathcal{P}(G)$. In fact, circuit P_1 must exactly finish at edge e_1 , because for any edge the rule of “stay on the same side” uniquely determines the edge before and after in the circuit.

Now select some edge f_1 of G that was not in P_1 , and repeat the process starting at f_1 . The new circuit P_2 will not contain an edge of P_1 by the same “stay on the same side” rule. Thus we obtain the next circuit in the partition. Repeat until all edges belong to some circuit. Since there is never a choice about which next edge to take, the partition is unique. Each edge is visited exactly once resulting in a linear runtime. \square

We call the circuits in $\mathcal{P}(G)$ the *osculating circuits* of G , see also Fig. 1(c). We distinguish two cases of $\mathcal{P}(G)$ based on whether or not the osculating circuits are simple directed cycles. It turns out that when a circuit visits a vertex twice, $\mathcal{P}(G)$ has a trivial structure.

Lemma 3. *Presume (C1–C3) hold. If some osculating circuit $P \in \mathcal{P}(G)$ visits a vertex twice, then P is the only element in $\mathcal{P}(G)$.*

Proof. Follow $P_1 \in \mathcal{P}(G)$, a non-simple osculating directed circuit, until the first time a vertex $v \in P_1$ is reached for the second time. P_1 can be partitioned into a simple directed cycle C' (by taking the part from v to v that we just followed) and a directed circuit C'' (the rest of P_1). Both C' and C'' are incident to v .

Fix an arbitrary drawing of G on the torus \mathcal{T} . By (C3) C' is non-contractible, so cutting \mathcal{T} along C' produces a cylinder. The cut will split v into two vertices, v' and v'' , one on each boundary of the cylinder.

Because of the osculating construction of P_1 , C' and C'' must intersect transversely at v (otherwise, P_1 would have terminated the first time it returned to v). Thus C'' contains an incident edge at each of the two copies, v' and v'' , of v . Taking the subpath D of C'' between v' and v'' , we obtain a path that travels on the cylinder from one boundary to the other. Cutting along D will cut the cylinder into one or more disks.

Now consider some other circuit $P_i \in \mathcal{P}(G)$, $P_i \neq P_1$. It cannot have a transverse crossing with either C' or C'' , because P_1 and P_i are osculating circuits. Therefore, it intersects neither C' nor D and we can conclude that P_i resides entirely within (or on the boundary) of one of the disks of $\mathcal{T} - C' - D$. But then P_i is a contractible directed circuit, a contradiction. So $\mathcal{P}(G)$ contains no osculating circuits other than P_1 . \square

The osculating partition can contain more than one element, see Fig. 1(c). It follows from the previous lemma that if $|\mathcal{P}(G)| > 1$, then all circuits in $\mathcal{P}(G)$ must be simple directed cycles.

In a transverse intersection under (C3'), if C_1 uses the left incoming (and right outgoing) edge of v we say that C_1 *crosses* C_2 *left-to-right* at v . Summing over all shared vertices, we define the *algebraic crossing number* of two circuits C_1 and C_2 to be:

$$\hat{i}(C_1, C_2) = \#\{C_1 \text{ crosses } C_2 \text{ left-to-right}\} - \#\{C_1 \text{ crosses } C_2 \text{ right-to-left}\}$$

Finally, using the following well known lemma, we can make a statement about the homotopy class of the osculating circuits in $\mathcal{P}(G)$:

Lemma 4. [9, p. 209] *Two closed simple curves C, C' have $\hat{i}(C, C') = 0$ if and only if they belong to the same homotopy class.*

Lemma 5. *If (C1–C3) hold, then all directed cycles $P_i \in \mathcal{P}(G)$ belong to the same homotopy class.*

Proof. If $\mathcal{P}(G)$ contains a non-simple circuit then, by Lemma 3, it is the only member of $\mathcal{P}(G)$ and the claim holds trivially. Otherwise, the osculating circuits of $\mathcal{P}(G)$ are all simple closed curves. None of them intersect transversally, which means that $\hat{i}(P_i, P_j) = 0$ for all pairs of osculating circuits $i \neq j$. This proves the result by Lemma 4. \square

A (*canonical*) *polygonal schema* for a toroidal graph consists of two fundamental cycles (called the *meridian* M and the *longitude* L) such that M and L intersect in exactly one point. Cutting G along the edges of $M \cup L$ will result in a topological disk. A circuit C belongs to *homotopy class* (m, ℓ) (with respect to a fixed polygonal schema) if $\hat{i}(C, L) = m$ and $\hat{i}(C, M) = \ell$.

With these terms in place, we can now state the thread conservation property via the following constraint:

- C4. There exists a meridian M , a longitude L and a partition $\mathcal{P}(G)$ of edges into osculating directed circuits such that each circuit in the partition is in the $(1, 0)$ -homotopy class.

The $(1, 0)$ -homotopy class restriction ensures that at each upward repeat all thread pairs return to the same left-right starting position. The thread conservation property is impossible to formulate as a condition of the combinatorial embedding because the homotopy class is affected by how the graph is drawn on

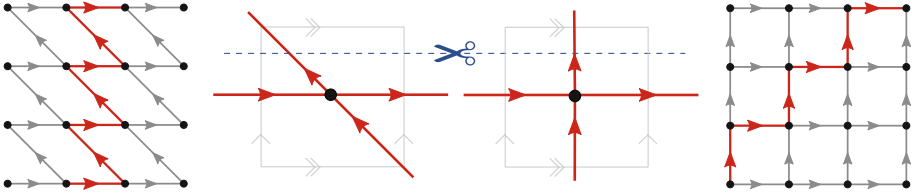


Fig. 2. A Dehn twist changes a valid lace graph on left into an invalid one on right. (Color figure online)

the torus. In particular, consider Fig. 2 which shows two drawings of the same graph on the torus differing by a homeomorphism known as a Dehn twist. Both drawings have the same combinatorial embedding, yet on the left side of Fig. 2 the red (bold) osculating circuit returns to its starting point while on the right side of the figure the osculating circuit drifts to the right.

Clearly, thread conservation demands that we fix more than the combinatorial embedding of the graph. However, based only on the combinatorial embedding, we can make a statement about the existence of a suitable graph drawing:

Lemma 6. *Given a digraph G that satisfies (C1–C3), there exists a drawing of G for which (C4) holds.*

Proof. By the Dehn-Lickorish theorem (see e.g. [5]) there exists a homeomorphism that maps any simple, non-contractible cycle to the $(1, 0)$ homotopy class of the torus. By Lemma 5, such a homeomorphism will map all elements in $\mathcal{P}(G)$ to the desired homotopy class. \square

The main contribution of this paper is a linear time algorithm for finding such a drawing:

Theorem 1. *Given a digraph G that satisfies (C1–C3), we can draw a lace pattern in linear time. The drawing resides in an $O(n^4) \times O(n^4)$ -grid.*

Proof. In Sect. 3, we provide an algorithm for finding a polygonal schema to satisfy (C4) for any digraph G that satisfies (C1–C3). We then use known algorithms ([1, 4]) for straight-line rectangular-frame drawings to create a lace pattern in the required time and space. \square

3 Finding a Polygonal Schema

In general, finding a polygonal schema with vertex-disjoint interiors is NP-hard [4]. However, for the purpose of drawing the lace-pattern, we do not need to find a polygonal schema within the given graph; it suffices (and in fact, is preferable) to add vertices and edges to the graph and find a polygonal schema within the additions. In this manner, the original edges of G are not on the schema boundary giving more freedom to where they can be placed. In this section we describe how to find such a supergraph with $O(n)$ nodes.

DrawLacePattern Algorithm

- A. Partition G into a set $\mathcal{P}(G)$ of osculating circuits and select one directed circuit P^* from the set.
- B. Create the offset graph $\mathcal{O}(G)$.
- C. Find a simple cycle M in $\mathcal{O}(G)$ such that $\hat{i}(M, P^*) = 0$ and M intersects every edge of G at most once.
- D. Find a simple cycle L in $\mathcal{O}(G)$ such that $\hat{i}(L, P^*) = \pm 1$, M and L intersect exactly once, and L intersects every edge of G at most once.
- E. Use existing torus-drawing techniques to draw G on a rectangle with meridian M and longitude L .

All steps are linear or constant time, and step E. will create a drawing of the required size, proving the theorem. Step A. was explained in Sect. 2.2 already; all other steps are explained below.

Creating the offset graph: We first create an offset graph $\mathcal{O}(G)$ in which we will choose a suitable meridian M and longitude L . Roughly speaking, $\mathcal{O}(G)$ is obtained by creating two copies of every osculating circuit P in $\mathcal{P}(G)$, which we will represent as \ddot{P} (used to find M) and \overline{P} (used to find L). In each copy, the circuits are separated (they do not share vertices as they do in G) and are simple cycles. The copies lie on top of G introducing crossings which we remove by inserting dummy vertices. Finally, for the simple cycles of \overline{P} , we connect the two halves of each vertex, split in the process of separating osculating paths, by introducing “crossover-edges”.

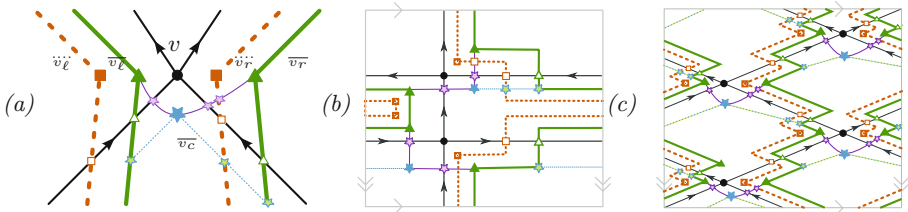


Fig. 3. Create offset graph. (a) Close-up near a vertex showing \ddot{P} (thick, dashed, square, orange), \overline{P} (thick, solid, triangle, green), crossover-edges (thin, solid, purple) and shortcut-edges (thin, dotted, blue). (b) Offset graph of non-simple osculating circuit. (c) Offset graph with multiple simple osculating cycles. (Color figure online)

Figure 3 illustrates two such toroidal graph embeddings $\mathcal{O}(G)$, with $O(n)$ vertices and edges, which we will now define formally. The *left-face* of a directed edge $v \rightarrow w$ is the face to the left of it while walking from v to w . For each vertex in $\mathcal{O}(G)$ we define $f_\ell(v)$ to be the left-face of the left incoming and left outgoing edges of v , $f_r(v)$ to be the right-face of the right incoming and right outgoing edges of v , and $f_c(v)$ to be the face incident to the left incoming and right incoming edges of v .

- Initially, $\mathcal{O}(G)$ contains all vertices and edges of G , embedded as in G .
- For every vertex $v \in V(G)$, add four new vertices $\ddot{v}_\ell, \overline{v}_\ell, \ddot{v}_r, \overline{v}_r$. Place $\ddot{v}_\ell, \overline{v}_\ell$ in $f_\ell(v)$ such that \overline{v}_ℓ is closer to the left incoming edge than \ddot{v}_ℓ . Place $\ddot{v}_r, \overline{v}_r$ in $f_r(v)$ such that \ddot{v}_r is closer to the right incoming edge than \overline{v}_r .
- For every edge $e = v \rightarrow w \in E(G)$, we add two new edges \ddot{e} and \bar{e} . If e is left outgoing at v , then \ddot{e} starts at \ddot{v}_ℓ and \bar{e} starts at \overline{v}_ℓ , else \ddot{e} starts at \ddot{v}_r and \bar{e} starts at \overline{v}_r . If e is left incoming at w , then \ddot{e} ends at \ddot{w}_ℓ and \bar{e} ends at \overline{w}_ℓ , else \ddot{e} ends at \ddot{w}_r and \bar{e} ends at \overline{w}_r .
- Note that edge $e \in E(G)$ may be intersected by its copies \ddot{e} and \bar{e} . This occurs, for example, when e is right outgoing at its tail and left incoming at its head. We remove these crossings (so that we again have a toroidal embedding) in the standard way by inserting dummy vertices that subdivide e, \ddot{e}, \bar{e} . (In the following descriptions, we will ignore these dummy-vertices, and speak of an edge e , even though it has become a path with 3 edges.)
- For every osculating circuit P there are now two circuits \ddot{P} and \bar{P} , using for each edge $e \in P$ the corresponding edges \ddot{e} and \bar{e} . Note that \ddot{P} and \bar{P} are simple, even if P is not. When P visits a vertex v twice, once via the incoming left and outgoing left edges of v and once via the incoming right and outgoing right edges of v , the corresponding \ddot{P} visits \ddot{v}_ℓ and \ddot{v}_r respectively and similarly \bar{P} visits \overline{v}_ℓ and \overline{v}_r . Due to the order of the copies near v , \ddot{P} and \bar{P} do not cross.
- Next add the *crossover-edge* $\bar{e}_v = (\overline{v}_\ell, \overline{v}_r)$ for each vertex $v \in V$. To obtain a toroidal embedding, route this edge so that it crosses three edges: the two incoming edges of G at v and the edge \ddot{e} that is incoming to \ddot{v}_r . These crossings are again replaced by dummy-vertices. In the crossover-edge insert a vertex v_c in the face $f_c(v)$.
- For a straight-line drawing, an edge e in $E(G)$ must not cross the rectangular frame twice. Consider an edge \bar{e} that crosses $e = v \rightarrow w$ where \bar{e} originates inside the face $f_c(w)$, say $\bar{e} = \overline{v}_r \rightarrow \overline{w}_\ell$. It may happen that the chosen longitude L contains \bar{e} followed by the crossover-edge $\bar{e}_w = (\overline{w}_\ell, \overline{w}_r)$ resulting in a double crossing of e by L . To avoid this situation, we add a *shortcut-edge* to $\mathcal{O}(G)$ that connects a point on \bar{e} inside $f_c(w)$ to w_c . Note that any circuit using $\bar{e}\bar{e}_w$ acts the same (with respect to algebraic crossing numbers) as a circuit shortened via the shortcut-edge, since all other crossings are unaffected.

Finding the meridian: In step A. we selected an osculating circuit P^* . Define M to be the copy \ddot{P}^* of circuit P^* in the offset graph (Fig. 4(a)). M is a simple cycle. It may cross P^* repeatedly but it does so only by switching back and forth between being left of P^* and right of P^* . In other words, $\hat{i}(P^*, M) = 0$ as desired. Finally M intersects every edge of G at most once by construction of \ddot{P} .

Finding the longitude: Define $\mathcal{L}(G)$ to be the subgraph of $\mathcal{O}(G)$ taking only the copies \bar{P} of osculating circuits, the crossover-edges, and the shortcut-edges. We claim that we can find a suitable longitude in $\mathcal{L}(G)$. The algorithm is quite simple (find a shortest path within $\mathcal{L}(G)$, with some edges removed), but arguing that it works is not.

Case (1) Circuit P^* : First consider the easier case in which P^* is not simple and is therefore, by Lemma 3, the only element in $\mathcal{P}(G)$. Let v be a vertex visited twice by P^* , hence $\overline{v_\ell}$ and $\overline{v_r}$ both belong to $\overline{P^*}$. Define L^* to be a subpath of $\overline{P^*}$ between $\overline{v_\ell}$ and $\overline{v_r}$, and \hat{L} to be L^* plus the crossover-edge $(\overline{v_r}, \overline{v_\ell})$. We have constructed $\overline{P^*}$ such that it does not intersect \check{P}^* . So L^* does not intersect M . The crossover-edge $(\overline{v_r}, \overline{v_\ell})$ intersects M exactly once. Therefore, \hat{L} intersects M exactly once as desired.

L^* and P^* may intersect numerous times between $\overline{v_\ell}$ and $\overline{v_r}$. But $\overline{v_\ell}$ is to the left of P^* while $\overline{v_r}$ is to the right, so in total L^* has one more left-to-right intersection than right-to-left-intersection. The crossover-edge $(\overline{v_r}, \overline{v_\ell})$ intersects both incoming edges at \overline{v} , and both of these edges belong to P^* . This adds two more right-to-left intersections, and so in total $i(\hat{L}, P^*) = -1$ as desired.

Case (2) Simple P^* : If P^* is simple then we find L using a path that connects “both sides” of P^* without crossing P^* . The existence of such a path is non-trivial, and crucially needs condition (C2), i.e., that the embedding is cellular. Specifically, we show:

Lemma 7. *Presume (C1–C3) hold. Let P be a directed osculating cycle. Then there exists a directed walk W in G that starts at a vertex $v \in P$ with a left outgoing edge $e_1 \notin P$, ends at a vertex $w \in P$ with a right incoming edge $e_k \notin P$, and has no transverse intersection or shared edges with P .*

Proof. (Sketch) The edge e_1 must exist otherwise the directed cycle P would bound a face, contradicting (C2) or (C3). One can now argue that starting from e_1 and always taking the left outgoing edge, we must reach such an edge e_k or find a contradiction to (C2). Details are in the full version. \square

Let Q be such a walk in G from $v \in P^*$ to $w \in P^*$, shortened by eliminating directed cycles (if any) so that it becomes a simple path. We obtain the longitude L by “translating” Q into $\mathcal{L}(G)$ and adding a subpath of $\overline{P^*}$ (Fig. 4(b) and (c)).

Note that the directed edges of Q need not be rotationally consecutive. For example, $e_i \in E(Q)$ may be right incoming at its head vertex x while $e_{i+1} \in E(Q)$ may be left outgoing at its tail vertex x . If $x \notin P^*$ then we can add the crossover-edge $(\overline{x_\ell}, \overline{x_r})$ to connect $\overline{e_i}$ and $\overline{e_{i+1}}$ without crossing $M = \check{P}^*$. If $x \in P^*$ but $e_i \notin P^*$ and $e_{i+1} \notin P^*$ then no crossover-edge is required because e_i and e_{i+1} are rotationally consecutive. It is not possible to have $x \in P^*$ and only one of $e_i \in P^*$ or $e_{i+1} \in P^*$ because of the way in which Q is defined.

Formally, let $Q = e_1, \dots, e_k$ be a simple path in $G - E(P^*)$ from $v \in P^*$ to $w \in P^*$ and let $L^- = \overline{e_1}, \dots, \overline{e_k}$ be a simple path using the corresponding edges in $\mathcal{L}(G)$ and crossover-edges as needed. L^- begins at $\overline{v_\ell}$ since e_1 is left outgoing and ends at $\overline{w_r}$ since e_k is right incoming. Define L^* to be the subpath of $\overline{P^*}$ connecting $\overline{w_\ell}$ to $\overline{v_r}$. Finally, define \hat{L} to be the simple cycle consisting of L^- , crossover-edge $(\overline{w_r}, \overline{w_\ell})$, L^* and crossover-edge $(\overline{v_r}, \overline{v_\ell})$.

Note that L^- and L^* never cross the meridian $M = \check{P}^*$. The only place where \hat{L} can intersect M is at the two crossover-edges $(\overline{v_r}, \overline{v_\ell})$ and $(\overline{w_r}, \overline{w_\ell})$. We claim that exactly one of them intersects \check{P}^* . To see this, recall that P^* uses the

right incoming and outgoing edge at v and the left incoming and outgoing edge at w . The order of vertices in the vicinity of v is $\ddot{v}_\ell, \overline{v}_\ell, v, \ddot{v}_r, \overline{v}_r$, and \dot{P}^* uses \ddot{v}_r , so $(\overline{v}_r, \overline{v}_\ell)$ crosses \dot{P}^* . On the other hand the order of vertices in the vicinity of w is $\ddot{w}_\ell, \overline{w}_\ell, w, \ddot{w}_r, \overline{w}_r$, and \dot{P}^* uses \ddot{w}_ℓ , so $(\overline{w}_r, \overline{w}_\ell)$ does not cross \dot{P}^* . Therefore, L and M cross exactly once as desired.

To see that $\hat{i}(\hat{L}, P^*) = \pm 1$, observe that L^- has no transverse intersections with P^* and so contributes nothing. L^* may intersect P^* repeatedly, alternatingly from left to right and from right to left, however, L^* starts on the left side of P^* at \overline{w}_ℓ and ends on the right side of P^* at \overline{v}_r , so it has a net of one left to right crossing. The crossover-edges $(\overline{w}_r, \overline{w}_\ell)$ and $(\overline{v}_r, \overline{v}_\ell)$ are both right to left crossings. Thus \hat{L} has one more crossing from right to left than it has crossings from left to right, yielding $\hat{i}(\hat{L}, P^*) = -1$ as desired.

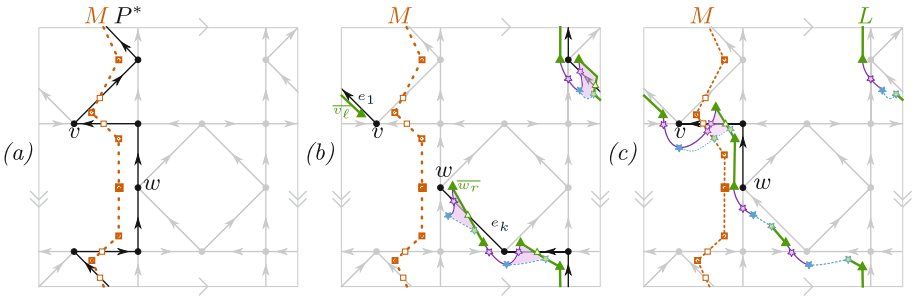


Fig. 4. (a) M is a shifted copy of P^* . (b) L exits from the left side of M at e_1 and returns on the right side of M at e_k . Loops where shortcuts are required are shaded purple. (c) Connect e_1 to e_k following P^* . (Color figure online)

Applying shortcuts: As desired, the simple cycle \hat{L} intersects M once and has $\hat{i}(\hat{L}, P^*) = \pm 1$. However, \hat{L} may intersect an edge e of G repeatedly. This happens only when \bar{e} transversely crosses e , and a crossover-edge at the tail of \bar{e} follows immediately after. Let L be the simple cycle obtained from \hat{L} by substituting the shortcut-edge at any such edge e . This has the same algebraic crossing number with P^* , still crosses M exactly once, and is a suitable longitude.

We finally remark that the simplest way to find L is as follows. Start with graph $\mathcal{L}(G)$ and remove all dummy-vertices that lie on \dot{P}^* . This effectively disallows any path in $\mathcal{L}(G)$ that crosses M . Pick any vertex $v \in P^*$ and, in what remains of $\mathcal{L}(G)$, find a shortest path L^- from \overline{v}_ℓ to \overline{v}_c (the vertex within the crossover-edge $(\overline{v}_r, \overline{v}_\ell)$). Finally add the edge $(\overline{v}_c, \overline{v}_\ell)$ to close the cycle into a suitable longitude L .

Drawing the graph: Let $\mathcal{T}(G)$ be the graph obtained from $\mathcal{O}(G)$ by removing all edges that do not belong to G, M or L and smoothing any degree-2 vertices. $\mathcal{T}(G)$ has a natural embedding on a torus with a well defined meridian and longitude. Cut $\mathcal{T}(G)$ along L to form graph $\mathcal{C}(G)$ which has a natural embedding on a cylinder with top and bottom boundaries L_t and L_b . Cut $\mathcal{C}(G)$

along M to obtain a planar graph $\mathcal{R}(G)$ with a natural embedding on a rectangle whose top/right/bottom/left sides are formed by the four boundary-paths $L_t/M_r/L_b/M_\ell$.

In our construction, we have been careful to ensure that boundary-cycles do not cross an edge of G more than once. As a consequence, we can now apply known algorithms for straight-line rectangular-frame drawings to $\mathcal{R}(G)$. The first such algorithm was given by Duncan et al. [4] and creates a drawing in an $O(n) \times O(n^2)$ -grid in linear time. Unfortunately, the algorithm does not necessarily produce a periodic drawing. We therefore turn to the drawing algorithm of Aleardi et al. [2], a modification of the algorithm of Duncan et al. which ensures periodic drawings. This is still achieved in linear time but at the cost of increasing the grid-size to $O(n^4) \times O(n^4)$ -grid. This proves Theorem 1.

4 Discussion and Open Problems

In this paper we provide an algorithm to generate a lace pattern given a directed graph and a fixed combinatorial embedding that satisfy the conditions (C1–C3). Our main challenge was to produce a graph drawing such that the threads in the pattern weave back and forth within a fixed width (follow a directed circuit that wraps once around the minor meridian axis and zero times around the major longitude axis). It turns out that if (C1–C3) are satisfied we can easily define a supergraph of linear size supporting a suitable rectangular schema and extract the schema in linear time. With care, we can sure that neither the meridian nor longitude of the schema intersects an edge of the graph twice, allowing us to reuse existing graph drawing techniques.

Our drawings are still somewhat unsatisfying for use as lace patterns. Consider an edge $e \in E(G)$ that crosses the rectangular schema, say the horizontal border L . The edge is broken into two parts e_t and e_b , where e_t intersects the top of the rectangle along L_t and e_b intersects the bottom along L_b . The algorithm of Aleardi et al. [2] ensures that the end points of e_t and e_b on L have the same x -coordinate but for e to be truly seen as “one edge”, e_t and e_b should also have the same slope. The bend is clearly visible when viewing several tiled repeats of the pattern as shown in Fig. 5. At the current time, we cannot see a way to simultaneously satisfy the topological $(1, 0)$ homotopy constraint enforced by the rectangular representation and provide smooth continuity of edges across this rectangular boundary.

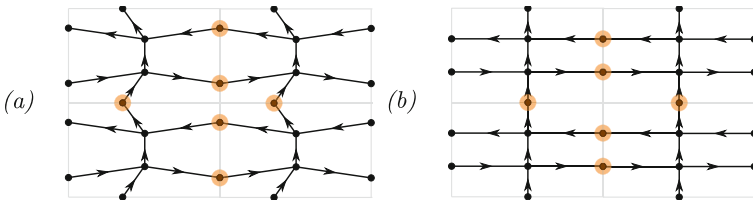


Fig. 5. Unwanted bends along border. (a) Result of algorithm. (b) Desired result.

References

1. Castelli Aleardi, L., Devillers, O., Fusy, É.: Canonical ordering for triangulations on the cylinder, with applications to periodic straight-line drawings. In: Didimo, W., Patrignani, M. (eds.) GD 2012. LNCS, vol. 7704, pp. 376–387. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36763-2_34
2. Castelli Aleardi, L., Fusy, É., Kostygin, A.: Periodic planar straight-frame drawings with polynomial resolution. In: Pardo, A., Viola, A. (eds.) LATIN 2014. LNCS, vol. 8392, pp. 168–179. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54423-1_15
3. Belcastro, S.M., Yackel, C.: Making Mathematics with Needlework: Ten Papers and Ten Projects. AK Peters, Natick (2008)
4. Duncan, C.A., Goodrich, M.T., Kobourov, S.G.: Planar drawings of higher-genus graphs. *J. Graph Algorithms Appl.* **15**(1), 7–32 (2011)
5. Farb, B., Margalit, D.: A Primer on Mapping Class Groups. Princeton University Press, Princeton (2011)
6. Grishanov, S., Meshkov, V., Omelchenko, A.: A topological study of textile structures. part I: an introduction to topological methods. *Text. Res. J.* **79**(8), 702–713 (2009)
7. Irvine, V., Ruskey, F.: Developing a mathematical model for bobbin lace. *J. Math. Arts* **8**(3–4), 95–110 (2014)
8. Mohar, B., Thomassen, C.: Graphs on Surfaces. John Hopkins University Press, Baltimore (2001)
9. Stillwell, J.: Classical Topology and Combinatorial Group Theory. GTM, vol. 72. Springer-Verlag, New York (1980)

Many Touchings Force Many Crossings

János Pach^{1,2} and Géza Tóth^{2(✉)}

¹ École Polytechnique Fédérale de Lausanne, St. 8, 1015 Lausanne, Switzerland

² Rényi Institute, Hungarian Academy of Sciences,
POB 127, Budapest 1364, Hungary
pach@cims.nyu.edu, geza@renyi.hu

Abstract. Given n continuous open curves in the plane, we say that a pair is *touching* if they have only one interior point in common and at this point the first curve does not get from one side of the second curve to its other side. Otherwise, if the two curves intersect, they are said to form a *crossing* pair. Let t and c denote the number of touching pairs and crossing pairs, respectively. We prove that $c \geq \frac{1}{10^5} \frac{t^2}{n^2}$, provided that $t \geq 10n$. Apart from the values of the constants, this result is best possible.

Keywords: Planar curves · Touching · Crossing

1 Introduction

In the context of the theory of topological graphs and graph drawing, many interesting questions have been raised concerning the adjacency structure of a family of curves in the plane or in another surface [5]. In particular, during the past four decades, various important properties of string graphs (i.e., intersection graphs of curves in the plane) have been discovered, and the study of different crossing numbers of graphs and their relations to one another has become a vast area of research. A useful tool in these investigations is the so-called crossing lemma of Ajtai, Chvátal, Newborn, Szemerédi and Leighton [1, 7]. It states the following: Given a graph of n vertices and $e > 4n$ edges, no matter how we draw it in the plane by not necessarily straight-line edges, there are at least constant times e^3/n^2 crossing pairs of edges.

This lemma has inspired a number of results establishing the existence of many crossing subconfigurations of a given type in sufficiently rich geometric or topological structures [2, 6, 10–12].

In this note, we will be concerned with families of curves in the plane. By a *curve*, we mean a non-selfintersecting continuous arc in the plane, that is, a homeomorphic image of the open interval $(0, 1)$. Two curves are said to *touch* each other if they have precisely one interior point in common and at this point the first curve does not pass from one side of the second curve to the other. Any other pair of curves with nonempty intersection is called *crossing*. A family of

curves is in *general position* if any two of them intersect in a finite number of points and no three pass through the same point.

Let n be even, t be a multiple of n , and suppose that $n \leq t < \frac{n^2}{4}$. Consider a collection A of $n - \frac{2t}{n} > \frac{n}{2}$ pairwise disjoint curves, and another collection B of $\frac{2t}{n}$ curves such that

- (i) $A \cup B$ is in general position,
- (ii) each element of B touches precisely $\frac{n}{2}$ elements of A , and
- (iii) no two elements of B touch each other.

The family $A \cup B$ consists of n curves such that the number of touching pairs among them is t . The only pairs of curves that may cross each other belong to B . Thus, the number of crossing pairs is at most $\binom{2t/n}{2} \leq \frac{2t^2}{n^2}$. See Fig. 1.

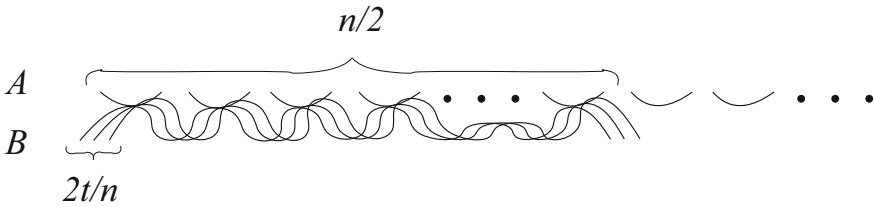


Fig. 1. A set of n curves with t touching pairs and at most $\frac{2t^2}{n^2}$ crossing pairs.

The aim of the present note is to prove that this construction is optimal up to a constant factor, that is, any family of n curves and t touchings has at least constant times $\frac{t^2}{n^2}$ crossing pairs.

Theorem 1. *Consider a family of n curves in general position in the plane which determines t touching pairs and c crossing pairs.*

If $t \geq 10n$, then we have $c \geq \frac{1}{10^8} \frac{t^2}{n^2}$. This bound is best possible up to a constant factor.

We make no attempt to optimize the constants in the theorem.

Pach et al. [8] established a similar relationship between t , the number of touching pairs, and C , the number of crossing *points* between the curves. They proved that $C \geq t(\log \log(t/n))^\delta$, for an absolute constant $\delta > 0$. Obviously, we have $C \geq c$. There is an arrangement of n red curves and n blue curves in the plane such that every red curve touches every blue curve, and the total number of crossing points is $C = \Theta(n^2 \log n)$; cf. [4]. Of course, the number of crossing pairs, c , can never exceed $\binom{n}{2}$.

Between n arbitrary curves, the number of touchings t can be as large as $(\frac{3}{4} + o(1))\binom{n}{2}$; cf. [9]. However, if we restrict our attention to algebraic plane curves of bounded degree, then we have $t = O(n^{3/2})$, where the constant hidden in the notation depends on the degree [3].

2 Proof of Theorem

We start with an easy observation.

Lemma. *Given a family of $n \geq 3$ curves in general position in the plane, no two of which cross, the number of touchings, t , cannot exceed $3n - 6$.*

Proof. Pick a different point on each curve. Whenever two curves touch each other at a point p , connect them by an edge (arc) passing through p . In the resulting drawing, any two edges that do not share an endpoint are represented by disjoint arcs. According to the Hanani-Tutte theorem [13], this means that the underlying graph is planar, so that its number of edges, t , satisfies $t \leq 3n - 6$. \square

Proof of Theorem. We proceed by induction on n . For $n \leq 20$, the statement is void. Suppose that $n > 20$ and that the statement has already been proved for all values smaller than n .

We distinguish two cases.

CASE A: $t \leq 10n^{3/2}$.

In this case, we want to establish the stronger statement

$$c \geq \frac{1}{10^4} \frac{t^2}{n^2}.$$

By the assumption, we have

$$\frac{1}{10^4} \frac{t^2}{n^2} \leq \frac{n}{100}. \tag{1}$$

Let G_t (resp., G_c) denote the *touching graph* (resp., *crossing graph*) associated with the curves. That is, the vertices of both graphs correspond to the curves, and two vertices are connected by an edge if and only if the corresponding curves are touching (resp., crossing).

Let T be a minimal vertex cover in G_c , that is, a smallest set of vertices of G_c such that every edge of G_c has at least one endpoint in T . Let $\tau = |T|$. Let U denote the complement of T . Obviously, U is an *independent set* in G_c . According to the Lemma, the number of edges in $G_t[U]$, the touching graph induced by U , satisfies

$$|E(G_t[U])| < 3|U| \leq 3n. \tag{2}$$

By the minimality of T , G_c has at least $|T| = \tau$ edges. That is, we have $c \geq \tau$, so we are done if $\tau \geq \frac{1}{10^4} \frac{t^2}{n^2}$.

From now on, we can and shall assume that $\tau < \frac{1}{10^4} \frac{t^2}{n^2}$. By (1), we have $\frac{1}{10^4} \frac{t^2}{n^2} \leq \frac{n}{100}$. Hence, $|T| \leq \frac{n}{100}$ and

$$|U| = n - |T| \geq \frac{99n}{100}. \tag{3}$$

Let $U' \subseteq U$ denote the set of all vertices in U that are not isolated in the graph G_c . By the definition of T , all neighbors of a vertex $v \in U$ in G_c belong to T . If $|U'| \geq \frac{1}{10^4} \frac{t^2}{n^2}$, then we are done, because $c \geq |U'|$.

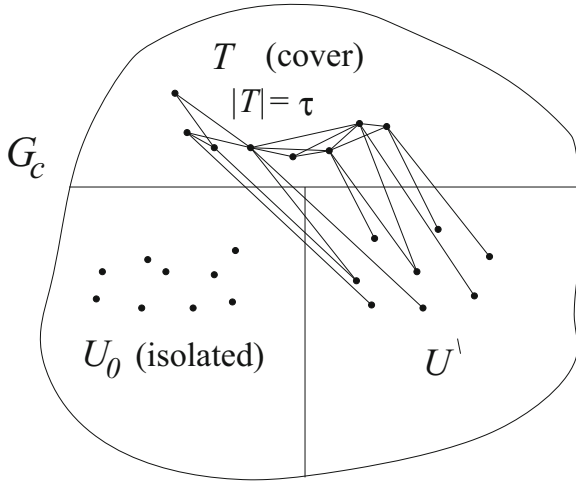


Fig. 2. Graph \$G_c\$.

Therefore, we can assume that

$$|U'| < \frac{1}{10^4} \frac{t^2}{n^2} \leq \frac{n}{100}, \tag{4}$$

where the second inequality follows again by (1) (Fig. 2).

Letting \$U_0 = U \setminus U'\$, by (3) and (4) we obtain \$|U_0| = |U| - |U'| \ge \frac{98n}{100}\$. Clearly, all vertices in \$U_0\$ are isolated in \$G_c\$.

Suppose that \$G_t[T \cup U']\$ has at least \$\frac{t}{10}\$ edges. Consider the set of curves \$T \cup U'\$. We have \$n_0 = |T \cup U'| \le \frac{2n}{100}\$ and, the number of touchings, \$t_0 = |E(G_t[T \cup U'])| \ge \frac{t}{10}\$. Therefore, by the induction hypothesis, for the number of crossings we have \$c_0 = |E(G_c[T \cup U'])| \ge \frac{1}{10^5} \frac{t_0^2}{n_0^2} \ge \frac{1}{10^4} \frac{t^2}{n^2}\$ and we are done. Hence, we assume in the sequel that \$G_t[T \cup U']\$ has fewer than \$\frac{t}{10}\$ edges.

Consequently, for the number of edges in \$G_t\$ running between \$T\$ and \$U_0\$, we have

$$|E(G_t[T, U_0])| \geq t - |E(G_t[T \cup U'])| - |E(G_t[U_0 \cup U'])| \geq t - \frac{t}{10} - 3n > \frac{t}{2}. \tag{5}$$

Here we used the assumption that \$t \ge 10n\$.

Let \$\chi = \chi(G_c[T])\$ denote the chromatic number of \$G_c[T]\$. In any coloring of a graph with the smallest possible number of colors, there is at least one edge between any two color classes. Hence, \$G_c[T]\$ has at least \$\binom{\chi}{2} \ge \frac{1}{10^4} \frac{t^2}{n^2}\$ edges, and we are done, provided that \$\chi > \frac{1}{70} \cdot \frac{t}{n}\$.

Thus, we can suppose that

$$\chi = \chi(G_c[T]) \leq \frac{1}{70} \cdot \frac{t}{n}. \tag{6}$$

Consider a coloring of $G_c[T]$ with χ colors, and denote the color classes by I_1, I_2, \dots, I_χ . Obviously, for every j , $I_j \cup U_0$ is an independent set in G_c . Therefore, by the Lemma, $G_t[I_j \cup U_0]$ has at most $3n$ edges. Summing up for all j and taking (6) into account, we obtain

$$|E(G_t[T, U_0])| \leq \sum_{j=1}^{\chi} |E(G_t[I_j \cup U_0])| \leq \frac{1}{70} \cdot \frac{t}{n} 3n \leq \frac{t}{20},$$

contradicting (5). This completes the proof in CASE A.

CASE B: $t \geq 10n^{3/2}$. Set $p = \frac{10n^3}{t^2} \leq \frac{1}{10}$. Select each curve independently with probability p . Let \mathbf{n}' , \mathbf{t}' , and \mathbf{c}' denote the number of selected curves, the number of touching pairs, and the number of crossing pairs between them, respectively. Clearly,

$$E[\mathbf{n}'] = pn, \quad E[\mathbf{t}'] = p^2t, \quad E[\mathbf{c}'] = p^2c. \tag{7}$$

The number of selected curves, \mathbf{n}' , has binomial distribution, therefore,

$$\text{Prob}[|\mathbf{n}' - pn| > \frac{1}{4}pn] < \frac{1}{3}. \tag{8}$$

By Markov's inequality,

$$\text{Prob}[\mathbf{c}' > 3p^2c] < \frac{1}{3}. \tag{9}$$

Consider the touching graph G_t . Let d_1, \dots, d_n denote the degrees of the vertices of G_t , and let e_1, \dots, e_t denote its edges, listed in any order. We say that an edge e_i is *selected* (or belongs to the random sample) if both of its endpoints were selected. Let X_i be the *indicator variable* for e_i , that is,

$$X_i = \begin{cases} 1 & \text{if } e_i \text{ was selected,} \\ 0 & \text{otherwise.} \end{cases}$$

We have $E[X_i] = p^2$. Let $\mathbf{t}' = \sum_{i=1}^t X_i$. It follows by straightforward computation that for every i ,

$$\text{var}[X_i] = E[(X_i - E[X_i])^2] = p^2 - p^4,$$

If e_i and e_j have a common endpoint for some $i \neq j$, then

$$\text{cov}[X_i, X_j] = E[X_i X_j] - E[X_i]E[X_j] = p^3 - p^4.$$

If e_i and e_j do not have a common vertex, then X_i and X_j are independent random variables and $\text{cov}[X_i, X_j] = 0$. Therefore, we obtain

$$\begin{aligned} \sigma^2 &= \text{var}[\mathbf{t}'] = \sum_{i=1}^t \text{var}[X_i] + \sum_{1 \leq i \neq j \leq t} \text{cov}[X_i, X_j] \\ &= (p^2 - p^4)t + (p^3 - p^4) \sum_{i=1}^n d_i(d_i - 1) < p^2t + 2p^3nt. \end{aligned}$$

From here, we get $\sigma < \sqrt{p^2t} + \sqrt{2p^3nt} < p^2t = E[\mathbf{t}']$. By Chebyshev's inequality,

$$\text{Prob}[|\mathbf{t}' - p^2t| \geq \lambda\sigma] \leq \frac{1}{\lambda^2}.$$

Setting $\lambda = \frac{1}{4}$,

$$\text{Prob}[|\mathbf{t}' - p^2t| \geq \frac{p^2t}{4}] \leq \frac{1}{4^2} < \frac{1}{3}. \tag{10}$$

It follows from (8), (9), and (10) that, with positive probability, we have

$$|\mathbf{n}' - pn| \leq \frac{1}{4}pn, \quad \mathbf{c}' \leq 3p^2c, \quad |\mathbf{t}' - p^2t| \leq \frac{1}{4}p^2t. \tag{11}$$

Consider a fixed selection of n' curves with t' touching pairs and c' crossing pairs for which the above three inequalities are satisfied. Then we have

$$\begin{aligned} t' &\geq \frac{3}{4}p^2t = \frac{300}{4} \cdot \frac{n^6}{t^3}, \\ n' &\leq \frac{5}{4}pn = \frac{50}{4} \cdot \frac{n^4}{t^2}, \end{aligned}$$

and, hence,

$$t' \geq \frac{6n^2}{t}n' \geq 10n'. \tag{12}$$

On the other hand,

$$\begin{aligned} t' &\leq \frac{5}{4}p^2t = \frac{500}{4} \cdot \frac{n^6}{t^3}, \\ n' &\geq \frac{3}{4}pn = \frac{30}{4} \cdot \frac{n^4}{t^2}, \end{aligned}$$

so that

$$10(n')^{3/2} \geq 10 \cdot \frac{30^{3/2}}{4^{3/2}} \cdot \frac{n^6}{t^3} > t'. \tag{13}$$

According to (12) and (13), the selected family meets the requirements of the Theorem in CASE A. Thus, we can apply the Theorem in this case to obtain that $c' \geq \frac{1}{10^4} \frac{t'^2}{n'^2}$. In view of (11), we have

$$3p^2c \geq c', \quad t' \geq \frac{3}{4}p^2t, \quad n' \leq \frac{5}{4}pn.$$

Thus,

$$3p^2c \geq c' \geq \frac{1}{10^4} \frac{t'^2}{n'^2} \geq \frac{1}{10^4} \frac{(3p^2t/4)^2}{(5pn/4)^2} = \frac{1}{10^4} \left(\frac{3}{5}\right)^2 \frac{p^2t^2}{n^2}.$$

Comparing the left-hand side and the right-hand side, we conclude that

$$c \geq \frac{1}{10^5} \frac{t^2}{n^2},$$

as required. This completes the proof of the Theorem. \square

Acknowledgment. The work of János Pach was partially supported by Swiss National Science Foundation Grants 200021-165977 and 200020-162884. Géza Tóth's work was partially supported by the Hungarian National Research, Development and Innovation Office, NKFIH, Grant K-111827.

References

1. Ajtai, M., Chvátal, V., Newborn, M., Szemerédi, E.: Crossing-free subgraphs. In: Theory and Practice of Combinatorics. North-Holland Mathematics Studies, vol. 60, pp. 9–12, North-Holland, Amsterdam (1982)
2. Dey, T.K.: Improved bounds on planar k -sets and related problems. *Discrete Comput. Geom.* **19**, 373–382 (1998)
3. Ellenberg, J. S., Solymosi, J., Zahl, J.: New bounds on curve tangencies and orthogonalities. *Discrete Anal.*, Paper No. 18, 22 pp. (2016)
4. Fox, J., Frati, F., Pach, J., Pinchasi, R.: Crossings between curves with many tangencies. In: Rahman, M.S., Fujita, S. (eds.) WALCOM 2010. LNCS, vol. 5942, pp. 1–8. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11440-3_1
5. Fox, J., Pach, J.: A separator theorem for string graphs and its applications. *Comb. Probab. Comput.* **19**, 371–390 (2010)
6. García, A., Noy, M., Tejel, J.: Lower bounds on the number of crossing-free subgraphs of K_n . *Comput. Geom.* **16**(4), 211–221 (2000)
7. Leighton, T.: Complexity Issues in VLSI, Foundations of Computing Series. MIT Press, Cambridge (1983)
8. Pach, J., Rubin, N., Tardos, G.: Beyond the Richter-Thomassen Conjecture. In: Proceedings of 27th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, pp. 957–968. SIAM (2016)
9. Pach, J., Tóth, G.: How many ways can one draw a graph? *Combinatorica* **26**(5), 559–576 (2006)
10. Sharir, M.: The Clarkson-Shor technique revisited and extended. *Comb. Prob. Comput.* **12**(2), 191–201 (2003)
11. Solymosi, J., Tóth, C.D.: Distinct distances in the plane. *Discrete Comput. Geom.* **25**(4), 629–634 (2001)
12. Székely, L.A.: Crossing numbers and hard Erdős problems in discrete geometry. *Comb. Prob. Comput.* **6**(3), 353–358 (1997)
13. Tutte, W.T.: Toward a theory of crossing numbers. *J. Comb. Theory* **8**, 45–53 (1970)

Thrackles: An Improved Upper Bound

Radoslav Fulek¹(✉) and János Pach^{2,3}

¹ IST Austria, Am Campus 1, 3400 Klosterneuburg, Austria
radoslav.fulek@gmail.com

² École Polytechnique Fédérale de Lausanne,
Station 8, 1015 Lausanne, Switzerland
pach@cims.nyu.edu

³ Rényi Institute, Hungarian Academy of Sciences,
P.O. Box 127, Budapest 1364, Hungary

Abstract. A *thrackle* is a graph drawn in the plane so that every pair of its edges meet exactly once: either at a common end vertex or in a proper crossing. We prove that any thrackle of n vertices has at most $1.3984n$ edges. *Quasi-thrackles* are defined similarly, except that every pair of edges that do not share a vertex are allowed to cross an *odd* number of times. It is also shown that the maximum number of edges of a quasi-thrackle on n vertices is $\frac{3}{2}(n-1)$, and that this bound is best possible for infinitely many values of n .

1 Introduction

Conway's thrackle conjecture [8] is one of the oldest open problems in the theory of topological graphs. A *topological graph* is a graph drawn in the plane so that its vertices are represented by points and its edges by continuous arcs connecting the corresponding points so that (i) no arc passes through any point representing a vertex other than its endpoints, (ii) any two arcs meet in finitely many points, and (iii) no two arcs are tangent to each other. A *thrackle* is a topological graph in which any pair of edges (arcs) meet precisely once. According to Conway's conjecture, every thrackle of n vertices can have at most n edges. This is analogous to Fisher's inequality [3]: If every pair of edges of a hypergraph H have precisely one point in common, then the number of edges of H cannot exceed the number of vertices.

The first linear upper bound on the number of edges of a thrackle, in terms of the number of vertices n , was established in [6]. This bound was subsequently improved in [1, 4], with the present record, $1.4n$, held by Goddyn and Xu [5], which also appeared in the master thesis of the second author [9]. One of the aims of this note is to show that this latter bound is not best possible.

R. Fulek—Gratefully acknowledges support from Austrian Science Fund (FWF): M2281-N35.

J. Pach—Supported by Swiss National Science Foundation Grants 200021-165977 and 200020-162884.

Theorem 1. *Any thrackle on $n > 3$ vertices has at most $1.3984n$ edges.*

Several variants of the thrackle conjecture have been considered. For example, Ruiz-Vargas *et al.* [7] established a linear upper bound on the number of edges even if two edges are allowed to be *tangent* to each other. The notion of *generalized thrackles* was introduced in [6]: they are topological graphs in which any pair of edges intersect an *odd* number of times, where each point of intersection is either a common endpoint or a proper crossing. A generalized thrackle in which no two edges incident to the same vertex have any other point in common is called a *quasi-thrackle*. We prove the following.

Theorem 2. *Any quasi-thrackle on n vertices has at most $\frac{3}{2}(n - 1)$ edges, and this bound is tight for infinitely many values of n .*

The proof of Theorem 1 is based on a refinement of parity arguments developed by Lovász *et al.* [6], by Cairns and Nikolayevsky [1], and by Goddyn and Xu [5], and it heavily uses the fact that two adjacent edges cannot have any other point in common. Therefore, one may suspect, as the authors of the present note did, that Theorem 1 generalizes to quasi-thrackles. Theorem 2 refutes this conjecture.

2 Terminology

Given a topological graph G in the projective or Euclidean plane, if it leads to no confusion, we will make no distinction in notation or terminology between its vertices and edges and the points and arcs representing them. A topological graph with no crossing is called an *embedding*. A connected component of the complement of the union of the vertices and edges of an embedding is called a *face*. A *facial walk* of a face is a closed walk in G obtained by traversing a component of the boundary of F . (The boundary of F may consist of several components.) The same edge can be traversed by a walk at most twice; the *length* of the walk is the number of edges counted with multiplicities. The edges of a walk form its *support*.

A pair of faces, F_1 and F_2 , in an embedding are *adjacent* (or *neighboring*) if there exists at least one edge traversed by a facial walk of F_1 and a facial walk of F_2 . In a connected graph, the *size* of a face is the length of its (uniquely determined) facial walk. A face of size k (resp., at least k or at most k) is called a *k-face* (resp., k^+ -face and k^- -face).

A *cycle* of a graph G is a closed walk along edges of G without vertex repetition. (To emphasize this property, sometimes we talk about “simple” cycles.) A cycle of length k is called a *k-cycle*.

A simple closed curve on a surface is said to be *one-sided* if its removal does not disconnect the surface. Otherwise, it is *two-sided*. An embedding of a graph G in the projective plane is called a *parity embedding* if every odd cycle of G is one-sided and every even cycle of G is two-sided. In particular, in a parity embedding every face is of even size.

3 Proof of Theorem 1

For convenience, we combine two theorems from [2,6].

Corollary 1. *A graph G is a generalized thrackle if and only if G admits a parity embedding in the projective plane. In particular, any bipartite thrackle can be embedded in the (Euclidean) plane.*

Proof. If G is a non-bipartite generalized thrackle, then, by a result of Cairns and Nikolayevsky [2, Theorem 2], it admits a parity embedding in the projective plane.

On the other hand, Lovász et al. [6, Theorem 1.4] showed that a bipartite graph is a generalized thrackle if and only if it is planar, in which case it can be embedded in the projective plane so that every cycle is two-sided. ■

The proof of the next lemma is fairly simple and is omitted in this version.

Lemma 1. *A thrackle does not contain more than one triangle.*

Next, we prove Theorem 1 for triangle-free graphs. Our proof uses a refinement of the discharging method of Goddyn and Xu [5].

Lemma 2. *Any triangle-free thrackle on $n > 3$ vertices has at most $1.3984(n-1)$ edges.*

Proof. Since no 4-cycle can be drawn as a thrackle, the lemma holds for graphs with fewer than 5 vertices. We claim that a vertex-minimal counterexample to the lemma is (vertex) 2-connected. Indeed, let $G = G_1 \cup G_2$, where $|V(G_1) \cap V(G_2)| < 2 \leq |V(G_1)|, |V(G_2)|$. Suppose that $|V(G_1)| = n'$. By the choice of G , we have $|E(G)| = |E(G_1)| + |E(G_2)| \leq 1.3984(n' - 1) + 1.3984(n - n') = 1.3984(n - 1)$.

Thus, we can assume that G is 2-connected. Using Corollary 1, we can embed G as follows. If G is not bipartite, we construct a parity embedding of G in the projective plane. If G is bipartite, we construct an embedding of G in the Euclidean plane. Note that in both cases, the size of each face of the embedding is even.

The following statement can be verified by a simple case analysis. It was removed from the short version of this note.

Proposition 1. *In the parity embedding of a 2-connected thrackle in the projective plane, the facial walk of every 8^- -face is a cycle, that is, it has no repeated vertex.*

To complete the proof of Lemma 2, we use a discharging argument. Since G is embedded in the projective plane, by Euler’s formula we have

$$e + 1 \leq n + f \tag{1}$$

where f is the number of faces and e is the number of edges of the embedding.

We put a charge $d(F)$ on each face F of G , where $d(F)$ denotes the size of F , that is, the length of its facial walk. An edge is called *bad* if it is incident to a 6-face. Let F be an 8^+ -face. Through every bad edge uv of F , we discharge from its charge a charge of $1/6$ to the neighboring 6-face on the other side of uv .

We claim that every face ends up with a charge at least 7. Indeed, we proved in [4] that in a thrackle no pair of 6-cycles can share a vertex. By Proposition 1, G has no 8-face with 7 bad edges. Furthermore, every 8^- -face is a 6-face or an 8-face, since in a parity embedding there is no odd face, and 4-cycles are not thrackable.

Proposition 2. *Unless G has 12 vertices and 14 edges, no two 8^+ -faces that share an edge can end up with charge precisely 7.*

Proof. An 8-face F with charge 7 must be adjacent to a pair of 6-faces, F_1 and F_2 . By Proposition 1, the facial walks of F, F_1 , and F_2 are cycles. Since G does not contain a cycle of length 4, both F_1 and F_2 share three edges with F , or one of them shares two edges with F and the other one four edges. Hence, any 8-face F' adjacent to F shares an edge uv with F , whose both endpoints are incident to a 6-face. If F' has charge 7, both edges adjacent to uv along the facial walk F' must be incident to a 6-face. By the aforementioned result from [4], these 6-faces must be F_1 and F_2 . By Proposition 1, the facial walk of F' is an 8-cycle. Since F' shares 6 edges with F_1 and F_2 , we obtain that G has only 4 faces F, F', F_1 , and F_2 . ■

In the case where G has 12 vertices and 14 edges, the lemma is true. By Proposition 2, if a pair of 8^+ -faces share an edge, at least one of them ends up with a charge at least $43/6$. Let F be such a face. We can further discharge $1/24$ from the charge of F to each neighboring 8^+ -face. After this step, the remaining charge of F is at least $\frac{43}{6} - 3 \cdot \frac{1}{24} = 7 + \frac{1}{24}$, which is possibly attained only by an 8-face that shares 5 edges with 6-faces. Every 9^+ -face F' has charge at least $d(F') - \frac{d(F')}{6} \geq 7 + \frac{1}{2}$.

In the last discharging step, we discharge through each bad edge of an 8^+ -face an additional charge of $1/288$ to the neighboring 6-face. At the end, the charge of every face is at least $7 + \frac{1}{24} - 6 \cdot \frac{1}{288} = 7 + \frac{1}{48}$. Since the total charge $\sum_F d(F) = 2e$ has not changed during the procedure, we obtain $2e \geq (7 + \frac{1}{48})f$. Combining this with (1), we conclude that

$$e \leq \frac{7 + \frac{1}{48}}{5 + \frac{1}{48}} n - \frac{7 + \frac{1}{48}}{5 + \frac{1}{48}} \leq 1.3984(n - 1),$$

which completes the proof of Lemma 2. ■

Now we are in a position to prove Theorem 1.

Proof of Theorem 1. If G does not contain a triangle, we are done by Lemma 2. Otherwise, G contains a triangle T . We remove an edge of T from G and denote the resulting graph by G' . According to Lemma 1, G' is triangle-free. Hence, by Lemma 2, G' has at most $1.3984(n - 1)$ edges, and it follows that G has at most $1.3984(n - 1) + 1 < 1.3984n$ edges. ■

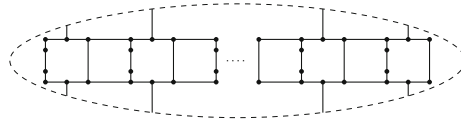


Fig. 1. Graph $H(k)$ embedded in the projective plane such that the embedding is a parity embedding. The projective plane is obtained by identifying the opposite pairs of points on the ellipse.

Remark 1. Without introducing any additional forbidden configuration, our methods cannot lead to an upper bound in Theorem 1, better than $\frac{22}{16}n = 1.375n$.

This is a simple consequence of the next lemma. Let $H(k)$ be a graph obtained by taking the union of a pair of vertex-disjoint paths $P = p_1 \dots p_{6k}$ and $Q = q_1 \dots q_{6k}$ of length $6k$; edges $p_i q_i$ for all $i \bmod 3 = 0$; edges $p_i q_{6k-i}$ for all $i \bmod 3 = 2$; and paths $p_i p'_i p''_i q_i$, for all $i \bmod 3 = 1$, which are internally vertex-disjoint from P, Q , and from one another.

Lemma 3. *For every $k \in \mathbb{N}$, the graph $H(k)$ has $16k$ vertices and $22k - 2$ edges, it contains no two 6-cycles that share a vertex or are joined by an edge, and it admits a parity embedding in the projective plane.*

Proof. For every k , $H(k)$ has $12k - 4$ vertices of degree three and $4k + 4$ vertices of degree two. Thus, $H(k)$ has $3(6k - 2) + 4k + 4 = 22k - 2$ edges. A projective embedding of $G(k)$ with the required property is depicted in Fig. 1. Using the fact that all 6-cycles are facial, the lemma follows. ■

Remark 2. It was stated without proof in [2] that the thrackle conjecture has been verified by computer up to $n = 11$. Provided that this is true, the upper bound in Theorem 1 can be improved to $e \leq \frac{7+\frac{1}{5}}{5+\frac{1}{5}}(n - 1) \leq 1.3847(n - 1)$. This follows from the fact that in this case an 8-face and a 6-face can share at most one edge and therefore we can maintain a charge of at least $7 + \frac{1}{5}$ on every face.

4 Proof of Theorem 2

It is known [1] that C_4 , a cycle of length 4, can be drawn as a generalized thrackle. Hence, our next result whose simple proof is left to the reader implies that the class of quasi-thracksles forms a proper subclass of the class of generalized thracksles.

Lemma 4. C_4 cannot be drawn as a quasi-thrackle.

Let $G(k)$ denote a graph consisting of k pairwise edge-disjoint triangles that intersect in a single vertex. The drawing of $G(3)$ as a quasi-thrackle, depicted in Fig. 2, can be easily generalized to any k . Therefore, we obtain the following

Lemma 5. *For every k , the graph $G(k)$ can be drawn as a quasi-thrackle.*

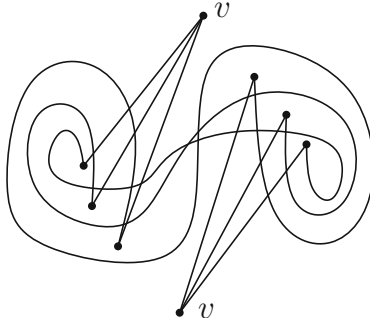


Fig. 2. A drawing of $G(3)$ as a quasi-thrackle. The two copies of the vertex v are identified in the actual drawing.

In view of Lemma 1, G_k cannot be drawn as a thrackle for any $k > 1$. Thus, the class of thrackles is a proper sub-class of the class of quasi-thrackles.

Cairns and Nikolayevsky [1] proved that every generalized thrackle of n vertices has at most $2n - 2$ edges, and that this bound cannot be improved. The graphs $G(k)$ show that for $n = 2k + 1$, there exists a quasi-thrackle with n vertices and with $\frac{3}{2}(n - 1)$ edges. According to Theorem 2, no quasi-thrackle with n vertices can have more edges.

Proof of Theorem 2. Suppose that the theorem is false, and let G be a counterexample with the minimum number n of vertices.

We can assume that G is 2-vertex-connected. Indeed, otherwise $G = G_1 \cup G_2$, where $|V(G_1) \cap V(G_2)| \geq 1$ and $E(G_1) \cap E(G_2) = \emptyset$. Suppose that $|V(G_1)| = n'$. By the choice of G , we have $|E(G)| = |E(G_1)| + |E(G_2)| \leq \frac{3}{2}(n' - 1) + \frac{3}{2}(n - n') = \frac{3}{2}(n - 1)$, so G is not a counterexample.

Suppose first that G is bipartite. By Corollary 1, G (as an abstract graph) can be embedded in the Euclidean plane. By Lemma 4, all faces in this embedding are of size at least 6. Using a standard double-counting argument, we obtain that $2e \geq 6f$, where e and f are the number of edges and faces of G , respectively. By Euler's formula, we have $e + 2 = n + f$. Hence, $6e + 12 \leq 6n + 2e$, and rearranging the terms we obtain $e \leq \frac{3}{2}(n - 6)$, contradicting our assumption that G was not a counterexample.

If G is not bipartite, then, according to Corollary 1, it has a parity embedding in the projective plane. By Lemma 4, G contains no 4-cycle. It does not have loops and multiple edges, therefore, the embedding has no 4-face. G cannot have a 5-face, because the facial walk of a 5-face would be either a one-sided 5-cycle (which is impossible), or it would contain a triangle and a cut-vertex (contradicting the 2-connectivity of G). The embedding of G also does not have a 3-face, since G is bipartite. By Euler's formula, $e + 1 = n + f$ and, as in the previous paragraph, we conclude that $6e + 6 \leq 6n + 2e$, the desired contradiction. ■

References

1. Cairns, G., Nikolayevsky, Y.: Bounds for generalized thrackles. *Discrete Comput. Geom.* **23**(2), 191–206 (2000)
2. Cairns, G., Nikolayevsky, Y.: Generalized thrackle drawings of non-bipartite graphs. *Discrete Comput. Geom.* **41**(1), 119–134 (2009)
3. Fisher, R.A.: An examination of the different possible solutions of a problem in incomplete blocks. *Ann. Hum. Genet.* **10**(1), 52–75 (1940)
4. Fulek, R., Pach, J.: A computational approach to Conway’s thrackle conjecture. *Comput. Geom.* **44**(67), 345–355 (2011)
5. Goddyn, L., Xu, Y.: On the bounds on Conway’s thrackles. *Discrete Comput. Geom.* **58**(2), 410–416 (2017)
6. Lovász, L., Pach, J., Szegedy, M.: On Conway’s thrackle conjecture. *Discrete Comput. Geom.* **18**(4), 369–376 (1997)
7. Ruiz-Vargas, A.J., Suk, A., Tóth, C.D.: Disjoint edges in topological graphs and the tangled-thrackle conjecture. *Eur. J. Combin.* **51**, 398–406 (2016)
8. Woodall, D.R.: Thrackles and deadlock. *Comb. Math. Appl.* **348**, 335–348 (1971)
9. Xu, Y.: Generalized thrackles and graph embeddings. M.Sc. thesis, Simon Fraser University (2014)

Orthogonal Representations and Book Embeddings

On Smooth Orthogonal and Octilinear Drawings: Relations, Complexity and Kandinsky Drawings

Michael A. Bekos, Henry Förster^(✉), and Michael Kaufmann

Wilhelm-Schickhard-Institut für Informatik, Universität Tübingen,
Tübingen, Germany
{bekos,foersth,mk}@informatik.uni-tuebingen.de

Abstract. We study two variants of the well-known orthogonal drawing model: (i) the smooth orthogonal, and (ii) the octilinear. Both models form an extension of the orthogonal, by supporting one additional type of edge segments (circular arcs and diagonal segments, respectively).

For planar graphs of max-degree 4, we analyze relationships between the graph classes that can be drawn bendless in the two models and we also prove NP-hardness for a restricted version of the bendless drawing problem for both models. For planar graphs of higher degree, we present an algorithm that produces bi-monotone smooth orthogonal drawings with at most two segments per edge, which also guarantees a linear number of edges with exactly one segment.

1 Introduction

Orthogonal graph drawing is an intensively studied and well established model for drawing graphs. As a result, several efficient algorithms providing good aesthetics and good readability have been proposed over the years, see e.g., [8, 18, 29, 35]. In such drawings, each vertex corresponds to a point on the Euclidean plane and each edge is drawn as a sequence of axis-aligned line segments; see Fig. 1.

Several research directions build upon this successful model. We focus on two models that have recently received attention: (i) the *smooth orthogonal* [5], in which every edge is a sequence of axis-aligned segments and circular arc segments with common axis-aligned tangents (i.e., quarter, half or three-quarter arc segments), and (ii) the *octilinear* [3], in which every edge is a sequence of axis-aligned and diagonal (at $\pm 45^\circ$) segments.

Observe that both models extend the orthogonal by allowing one more type of edge-segments. The former was introduced with the aim of combining the artistic appeal of *Lombardi drawings* [13, 15] with the clarity of the orthogonal drawings. The latter, on the other hand, is primarily motivated by metro-map and map schematization applications (see, e.g., [25, 31, 32, 34]). Note that in the orthogonal and in the smooth orthogonal models, each edge may enter a vertex

This work is supported by DFG grant Ka812/17-1.

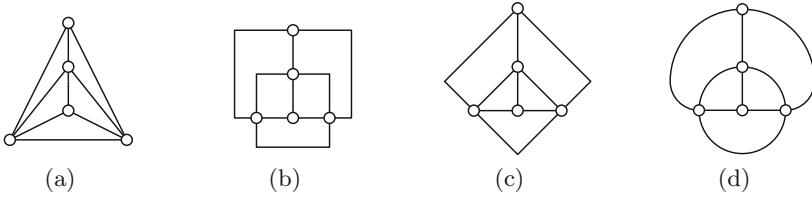


Fig. 1. Different drawings of a planar graph of max-degree 4: (a) straight-line, (b) orthogonal 3-drawing, (c) octilinear 2-drawing, and (d) smooth orthogonal 2-drawing.

using one out of four available (axis-aligned) directions, called *ports*. Thus both models support graphs of max-degree 4. In the octilinear model, each vertex has eight available ports and therefore one can draw graphs of max-degree 8.

For readability purposes, usually in such drawings one seeks to minimize the *edge complexity* [11, 27], i.e., the maximum number of segments used for representing any edge. Also, when the input is a planar graph, one seeks for a corresponding planar drawing. Note that drawings with edge complexity 1 are also called *bendless*. We refer to drawings with edge complexity k as k -drawings; thus, by definition, orthogonal k -drawings have at most $k - 1$ bends per edge.

Known results. There exists a plethora of results for each of the aforementioned models; here we list existing results for drawings with low edge complexity.

- All planar graphs of max-degree 4, except for the octahedron, admit orthogonal 3-drawings; the octahedron is orthogonal 4-drawable [8, 29]. Minimizing the number of bends over all embeddings of a planar graph of max-degree 4 is \mathcal{NP} -hard [22]. For a given planar embedding, however, finding a planar orthogonal drawing with minimum number of bends can be done in polynomial time by an approach, called *topology-shape-metrics* [35], that is based on min-cost flow computations and works in three phases. Initially, a planar embedding is computed if not specified by the input. In the next phase, the angles and the bends of the drawing are computed, yielding an *orthogonal representation*. In the last phase, the actual coordinates for the vertices and bends are computed.
- All planar graphs of max-degree 4 (including the octahedron) admit smooth orthogonal 2-drawings. Note that not all planar graphs of max-degree 4 allow for bendless smooth orthogonal drawings [5], and that such drawings may require exponential area [1]. Bendless smooth orthogonal drawings are possible only for subclasses, e.g., for planar graphs of max-degree 3 [4] and for outerplanar graphs of max-degree 4 [1]. It is worth mentioning that the complexity of the problem, whether a planar graph of max-degree 4 admits a bendless smooth orthogonal drawing, has not been settled (it is conjectured to be \mathcal{NP} -hard [1]).
- All planar graphs of max-degree 8 admit octilinear 3-drawings [28], while planar graphs of max-degree 4 or 5 allow for octilinear 2-drawings [3]. Bendless octilinear drawings are always possible for planar graphs of max-degree 3 [23].

Note that deciding whether an embedded planar graph of max-degree 8 admits a bendless octilinear drawing is \mathcal{NP} -hard [31]. It is not, however, known whether this negative result applies for planar graphs of max-degree 4 or whether these graphs allow for a decision algorithm (in fact, there exist planar graphs of max-degree 4 that do not admit bendless octilinear drawings [6]).

Our contribution. Motivated by the fact that usually one can “easily” convert an octilinear drawing of a planar graph of max-degree 4 to a corresponding smooth orthogonal one (e.g., by replacing diagonal edge segments with quarter circular arc segments; see Figs. 1c and d for an example), and vice versa, we study in Sect. 2 inclusion-relationships between the graph-classes that admit such drawings. In Sect. 3, we show that it is \mathcal{NP} -hard to decide whether an embedded planar graph of max-degree 4 admits a bendless smooth orthogonal or a bendless octilinear drawing, in the case where the angles between any two edges incident to a common vertex and the shapes of all edges are specified as part of the input (e.g., as in the last step of the topology-shape-metrics approach [35]). Our proof is a step towards settling the complexities of both decision problems in their general form. Inspired from the *Kandinsky model* (see, e.g., [7, 10, 18]) for drawing planar graphs of arbitrary degree in an orthogonal style, we present in Sect. 4 two drawing algorithms that yield bi-monotone smooth orthogonal drawings of good quality. The first yields drawings of smaller area, which can also be transformed to octilinear with bends at 135° . The second yields larger drawings but guarantees that at most $2n - 5$ edges are drawn with two segments. We conclude in Sect. 5 with open problems.

Preliminaries. For graph theoretic notions refer to [24]. For definitions on planar graphs, we point the reader to [11, 27]. We also assume familiarity with standard graph drawing techniques, such as the *canonical ordering* [19, 26] and the *shift-method* by de Fraysseix et al. [19]; see [2] for more details.

2 Relationships Between Graph Classes

In this section, we consider relationships between the classes of graphs that admit smooth orthogonal k -drawings and octilinear k -drawings, $k \geq 1$, denoted as SC_k and $8C_k$, respectively. Our findings are also summarized in Fig. 2.

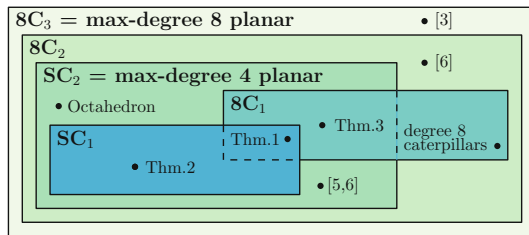


Fig. 2. Different inclusion-relationships: For $k \geq 1$, SC_k and $8C_k$ correspond to the classes of graphs admitting smooth orthogonal and octilinear k -drawings, respectively.

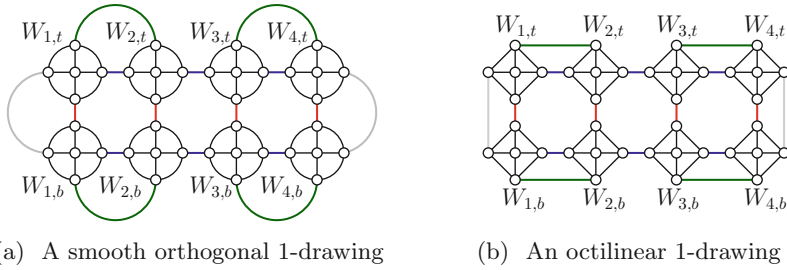


Fig. 3. Illustrations for the proof of Theorem 1.

By definition, $SC_1 \subseteq SC_2$ and $8C_1 \subseteq 8C_2 \subseteq 8C_3$ hold. Since each planar graph of max-degree 8 admits an octilinear 3-drawing [28], class $8C_3$ coincides with the class of planar graphs of max-degree 8. Similarly, class SC_2 coincides with the class of planar graphs of max-degree 4, as these graphs admit smooth orthogonal 2-drawings [1]. This also implies that $SC_2 \subseteq 8C_2$, since each planar graph of max-degree 4 admits an octilinear 2-drawing [3]. The relationship $8C_2 \neq 8C_3$ follows from [3], where it was proven that there exist planar graphs of max-degree 6 that do not admit octilinear 2-drawings. The relationship $SC_2 \neq 8C_2$ follows from [6], where it was shown that there exist planar graphs of max-degree 5 that admit octilinear 2-drawings and no octilinear 1-drawings, and the fact that planar graphs of max-degree 5 cannot be drawn in the smooth orthogonal model. The octahedron graph admits neither a bendless smooth orthogonal drawing [5] nor a bendless octilinear drawing [6]. However, since it is of max-degree 4, it admits 2-drawings in both models [1, 3]. Hence, it belongs to $8C_2 \cap SC_2 \setminus (8C_1 \cup SC_1)$. To prove that $8C_1 \setminus SC_2 \neq \emptyset$, observe that a caterpillar whose spine vertices are of degree 8 clearly admits an octilinear 1-drawing, however, due to its degree it does not admit a smooth orthogonal.

To complete the discussion of the relationships of Fig. 2, we have to show that SC_1 and $8C_1$ are incomparable. This is the most interesting part of our proof, as usually one can “easily” convert a bendless octilinear drawing of a planar graph of max-degree 4 to a corresponding bendless smooth orthogonal one (e.g., by replacing diagonal segments with quarter circular arcs), and vice versa; see, e.g., Figs. 1c and d. Since the endpoints of each edge of a bendless smooth orthogonal or octilinear drawing are along a line with slope 0, 1, -1 or ∞ , such conversions are in principle possible. Two difficulties that might arise are to preserve planarity and to guarantee that no two edges enter a vertex using the same port. Clearly, however, there exist infinitely many (even 4-regular) planar graphs that admit both drawings in both models; see Fig. 3 and [2] for more details.

Theorem 1. *There is an infinitely large family of 4-regular planar graphs that admit both bendless smooth orthogonal and bendless octilinear drawings.*

In the next two theorems we show that SC_1 and $8C_1$ are incomparable.

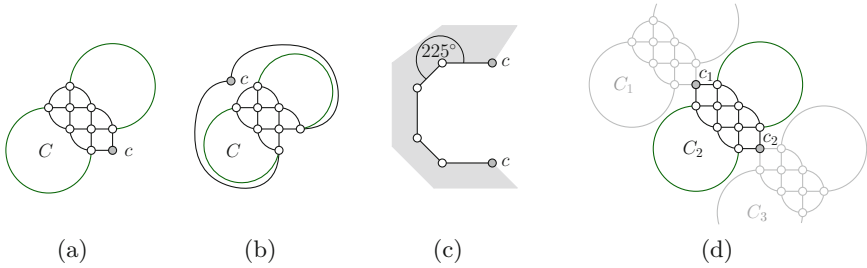


Fig. 4. Illustrations for the proof of Theorem 2.

Theorem 2. *There is an infinitely large family of 4-regular planar graphs that admit bendless smooth orthogonal drawings but no bendless octilinear drawings.*

Proof. Consider the planar graph C of Fig. 4a, which is drawn bendless smooth orthogonal. We claim that C admits no bendless octilinear drawing. If one substitutes its degree-2 vertex (denoted by c in Fig. 4a) by an edge connecting its two neighbors, then the resulting graph is triconnected, which admits a unique embedding (up to the choice of its outerface; see Figs. 4a and b). Now, observe that the outerface of any octilinear drawing of graph C (if any) has length at most 5 (Constraint 1). In addition, each vertex of this outerface (except for c , which is of degree 2) must have two ports pointing in the interior of this drawing, because every vertex of C is of degree 4 except for c . This implies that the angle formed by any two consecutive edges of this outerface is at most 225° , except for the pair of edges incident to c (Constraint 2). But if we want to satisfy both constraints, then at least one edge of this outerface must be drawn with a bend; see Fig. 4c. Hence, graph C does not admit a bendless octilinear drawing.

Based on graph C , for each $k \in \mathbb{N}_0$ we construct a 4-regular planar graph G_k consisting of $k + 2$ biconnected components C_1, \dots, C_{k+2} arranged in a chain; see Fig. 4d for the case $k = 1$. Clearly, G_k admits a bendless smooth orthogonal drawing for any k . Since the end-components of the chain (i.e., C_1 and C_{k+2}) are isomorphic to C , G_k does not admit a bendless octilinear drawing for any k . \square

Theorem 3. *There is an infinitely large family of 4-regular planar graphs that admit bendless octilinear drawings but no bendless smooth orthogonal drawings.*

Proof (sketch). Consider the planar graph B of Fig. 5a, which is drawn bendless octilinear. Graph B has two separation pairs (i.e., $\{t_1, t_2\}$ and $\{p_1, p_2\}$ in Fig. 5a).

Based on graph B , for each $k \in \mathbb{N}_0$ we construct a 4-regular planar graph G_k consisting of $2k + 4$ copies of B arranged in a cycle; see Fig. 5b where each copy of B is drawn as a gray-shaded parallelogram. By construction, G_k admits a bendless octilinear drawing for any k . By planarity at least one copy of graph B must be embedded with the outerface of Fig. 5a. However, if we require the outerface of B to be the one of Fig. 5a, then all possible planar embeddings of B are isomorphic to the one of Fig. 5a. We exploit this property in [2] to show that B does not admit a bendless smooth orthogonal drawing with this outerface. The

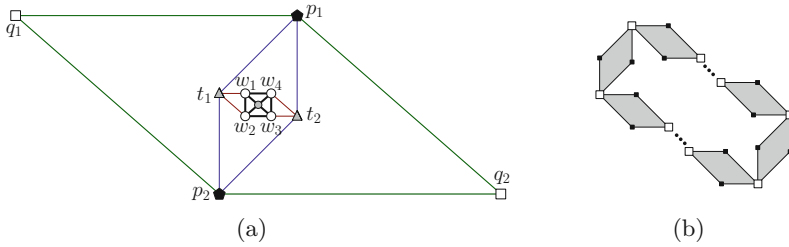


Fig. 5. Illustrations for the proof of Theorem 3.

detailed proof is based on an exhaustive consideration of all bendless smooth orthogonal drawings of subgraphs of B , which we incrementally augment by adding more vertices to them. Thus, for any k , graph G_k does not admit a bendless smooth orthogonal drawing. \square

3 \mathcal{NP} -hardness Results

In this section, we study the complexity of the bendless smooth orthogonal and octilinear drawing problems. As a first step towards addressing the complexity of both problems for planar graphs of max-degree 4 in general, here we make an additional assumption. We assume that the input, apart from an embedding, also specifies a *smooth orthogonal* or an *octilinear representation*, which are defined analogously to the orthogonal ones: (i) the angles between consecutive edges incident to a common vertex in the cyclic order around it (given by the planar embedding) are specified, and (ii) the *shape* of each edge (e.g., straight-line, or quarter-circular arc) is also specified. In other words, we assume that our input is analogous to the one of the last step of the topology-shape-metrics approach [35].

Theorem 4. *Given a planar graph G of max-degree 4 and a smooth orthogonal representation \mathcal{R} , it is \mathcal{NP} -hard to decide whether G admits a bendless smooth orthogonal drawing preserving \mathcal{R} .*

Proof. Our reduction is from the well-known 3-SAT problem [21]. Given a formula φ in conjunctive normal form, we construct a graph G_φ and a smooth orthogonal representation \mathcal{R}_φ , such that G_φ admits a bendless smooth orthogonal drawing Γ_φ preserving \mathcal{R}_φ if and only if φ is satisfiable; see also Fig. 6.

The main ideas of our construction are: (i) specific straight-line edges in Γ_φ transport *information* encoded in their length, (ii) rectangular faces of Γ_φ propagate the edge length of one side to its opposite, and (iii) for a face composed of two straight-line edges and a quarter circle arc, the straight-line edges are of same length, which allows us to change the *direction* in which the information “flows”.

Variable gadget. For each variable x of φ , we introduce a gadget; see Figs. 7a and b. The bold-drawn quarter circle arc ensures that the sum of the edge lengths to its left is the same as the sum of the edge lengths to its bottom (refer to the edges

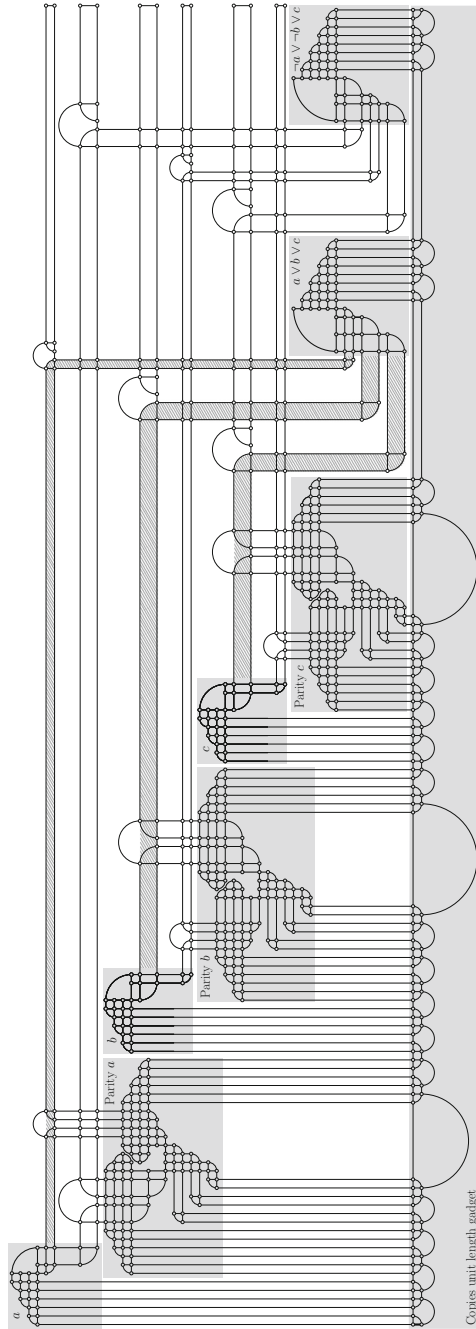


Fig. 6. Drawing Γ_φ for $\varphi = (a \vee b \vee c) \wedge (\bar{a} \vee \bar{b} \vee c)$ and the assignment $a = \text{false}$ and $b = c = \text{true}$.

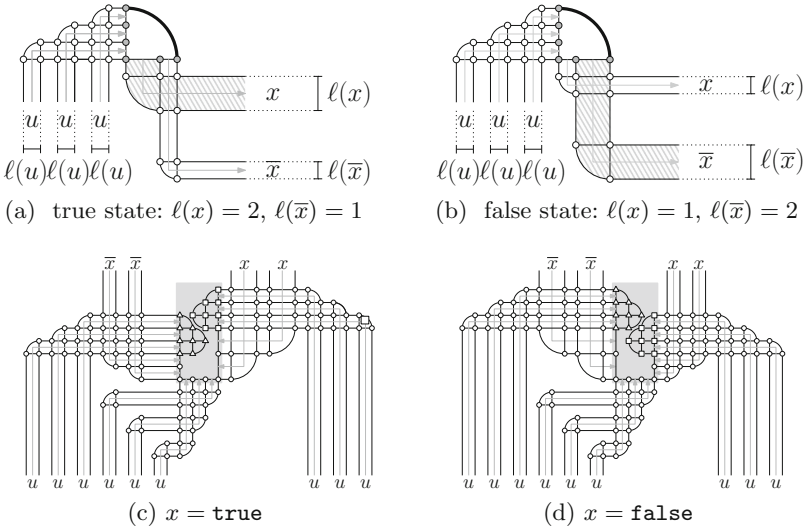


Fig. 7. In Figures (a) and (b) the variable gadget is illustrated. In Figures (c) and (d) the parity gadget is illustrated; gray-colored arrows show the information “flow”.

with gray endvertices). As “input” the gadget gets three edges of unit length $\ell(u)$. This ensures that $\ell(x) + \ell(\bar{x}) = 3 \cdot \ell(u)$ holds for the “output literals” x and \bar{x} , where $\ell(x)$ and $\ell(\bar{x})$ denote the lengths of two edges representing x and \bar{x} .

To introduce our concept, assume that the lengths of all straight-line edges are integral and at least 1. If we could require $\ell(u) = 1$, then $\ell(x), \ell(\bar{x}) \in \{1, 2\}$. This would allow us to encode the assignment $x = \text{true}$ with $\ell(x) = 2$ and $\ell(\bar{x}) = 1$, and the assignment $x = \text{false}$ with $\ell(x) = 1$ and $\ell(\bar{x}) = 2$. However, if we cannot avoid, e.g., that $\ell(u) = 2$, then the variable gadget would not prevent us from setting $\ell(x) = \ell(\bar{x}) = 3$, which means that x and \bar{x} are “half-true”. We solve this issue by the so-called *parity gadget*, that allows us to relax the integral constraint and to ensure that $\ell(x), \ell(\bar{x}) \in \{\ell(u) + \varepsilon, 2\ell(u) - \varepsilon\}$, for $0 < \varepsilon \ll \ell(u)$.

Parity gadget. For each variable x of φ , G_φ has a gadget (see Figs. 7c and d), which results in overlaps in Γ_φ , if the values of $\ell(x)$ and $\ell(\bar{x})$ do not differ significantly. The central part of this gadget is a “vertical gap” of width $3 \cdot \ell(u)$ (shaded in gray in Figs. 7c and d) with two blocks of vertices (triangular- and square-shaped in Figs. 7c and d) pointing inside the gap. Each block defines two square-shaped faces and three faces of length 3, each formed by two straight-line edges and a quarter circle arc. Depending on the choice of $\ell(x)$ and $\ell(\bar{x})$, one of the blocks may be located above the other. If $\ell(x) \approx \ell(\bar{x})$, however, we can observe that the two blocks are not far enough apart from each other, which leads to overlaps. Using elementary geometry, we prove in [2] that overlaps can be avoided if and only if $|\ell(\bar{x}) - \ell(x)| > \sqrt{3}/2 \cdot \ell(u) \approx 0.866 \cdot \ell(u)$, which implies: that $\ell(x), \ell(\bar{x}) \in (0, 1.067 \cdot \ell(u)] \cup [1.933 \cdot \ell(u), 3)$, i.e., $\varepsilon < 0.067 \cdot \ell(u)$.

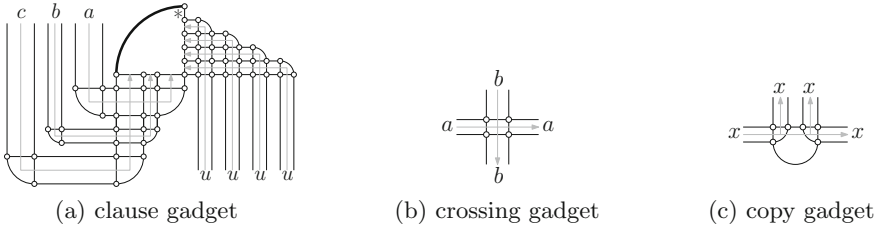


Fig. 8. Different gadgets; gray-colored arrows show the information “flow”.

Clause gadget. For each clause of φ with literals a , b and c , we introduce a gadget, which is illustrated in Fig. 8a. The bold-drawn quarter circle arc of Fig. 8a compares two sums of information. From the righthand side, four edges of unit length “enter” the arc. Observe that there is also a *free edge* (marked with an asterisk in Fig. 8a), which also contributes to the sum but can be stretched independently of any other edge. Hence, the sum of edge lengths on the righthand side of this arc is $>4 \cdot \ell(u)$. The three literals “enter” at the bottom; the sum here is $\ell(a) + \ell(b) + \ell(c)$. Combining both, we obtain that $\ell(a) + \ell(b) + \ell(c) > 4 \cdot \ell(u)$ must hold. This implies that not all a , b and c can be **false**, since in this case $\ell(a) + \ell(b) + \ell(c) = 3 \cdot (\ell(u) + \varepsilon) < 4 \cdot \ell(u)$.

Auxiliary gadgets. The *crossing gadget* just consists of a rectangle and is used to allow two flows of information to cross each other; see Fig. 8b. The *copy gadget* takes an information and creates three copies of this information; see Fig. 8c. This is because both quarter circular arcs of the copy gadget must have the same radius in the presence of the half circular arc of the copy gadget. Finally, the *unit length gadget* is a single edge, which we assume to be of length $\ell(u)$.

We now describe our construction; see Fig. 6: G_φ contains one unit length gadget, which is copied several times using the copy gadget (the number of copies depends linearly on the number of variables ν and clauses μ of φ). For each variable of φ , G_φ has a variable gadget and a parity gadget, each of which is connected to different copies of the unit length gadget. For each clause of φ , G_φ has a clause gadget, which has four connections to different copies of the unit length gadget. We compute \mathcal{R}_φ as follows. We place the variable gadget of each variable x above and to the left of its parity gadget and we connect the output literals of the variable gadget of x with its parity gadget through a copy gadget. We place the variable and the parity gadgets of the i -th variable below and to the right of the corresponding ones of the $(i - 1)$ -th variable. We place each clause gadget to the right of the sketch constructed so far, so that the gadget of the i -th clause is to the right of the $(i - 1)$ -th clause. This allows us to connect copies of the output literals of the variable gadget of each variable with the clause gadgets that contain it, so that all possible crossings (which are resolved using the crossing gadget) appear above the clause gadgets. More precisely, if a clause contains a literal of the i -th variable, we have a crossing with the literals of all variables with indices $(i + 1)$ to ν . Hence, for each clause we add $O(\nu)$ crossing

and three copy gadgets. Note that all copy gadgets of the unit length gadget lie below all variable, parity, and clause gadgets. The obtained representation \mathcal{R}_φ conforms with the one of Fig. 6. The construction can be done in $O(\nu\mu)$ time.

To complete the proof, assume that G_φ admits a bendless smooth orthogonal drawing Γ_φ preserving \mathcal{R}_φ . For each variable x of φ , we set x to **true** if and only if $\ell(x) \geq 1.933 \cdot \ell(u)$. Since for each clause $(a \vee b \vee c)$ of φ we have that $\ell(a) + \ell(b) + \ell(c) > 4 \cdot \ell(u)$, at least one of a , b and c must be **true**. Hence, φ admits a truth assignment. For the opposite direction, based on a truth assignment of φ , we can set, e.g., $\ell(x) = 1.95$ and $\ell(\bar{x}) = 1.05$ for each variable x , assuming that $\ell(u) = 1$. Then, arranging the variable and the clause gadgets of G_φ as in Fig. 6 yields a bendless smooth orthogonal drawing Γ_φ preserving \mathcal{R}_φ . \square

Remark 1. The special case of our problem, in which circular arcs are not present, is known as *HV-rectilinear planarity testing* [30]. As opposed to our problem, HV-rectilinear planarity testing is polynomial-time solvable in the fixed embedding setting [14] (and becomes \mathcal{NP} -hard in the variable embedding setting [12]).

Theorem 5. *Given a planar graph G of max-degree 4 and an octilinear representation \mathcal{R} , it is \mathcal{NP} -hard to decide whether G admits a bendless octilinear drawing preserving \mathcal{R} .*

Proof (sketch). Except for the parity gadget, we can adjust to the octilinear model simply by replacing arcs with diagonal segments; for details see [2]. In this case the parity gadget guarantees $|\ell(x) - \ell(\bar{x})| > 5/6 \cdot \ell(u) \approx 0.833 \cdot \ell(u)$, which implies that $\varepsilon < 0.084 \cdot \ell(u)$. \square

4 Bi-monotone Drawings

In this section, we study variants of the *Kandinsky* drawing model [7, 10, 18], which forms an extension of the orthogonal model to graphs of degree greater than 4. In this model, the vertices are represented as squares, placed on a *coarse grid*, with multiple edges attached to each side of them aligned on a *finer grid*.

The Kandinsky model allows for natural extensions to both smooth orthogonal and octilinear models. We are aware of only one preliminary result in this direction: A linear time drawing algorithm is presented in [5] for the production of smooth orthogonal 2-drawings for planar graphs of arbitrary degree in quadratic area, in which all vertices are on a line ℓ and the edges are drawn either as half circles (above or below ℓ), or as two consecutive half circles one above and one below ℓ (i.e., the latter ones are of complexity 2, but they are at most $n - 2$).

For an input maximal planar graph G (of arbitrary degree), our goal is to construct a smooth orthogonal (or an octilinear) 2-drawing for G with the following aesthetic benefits over the aforementioned drawing algorithm: (i) the vertices are distributed evenly over the drawing area, and (ii) each edge is *bi-monotone* [20],

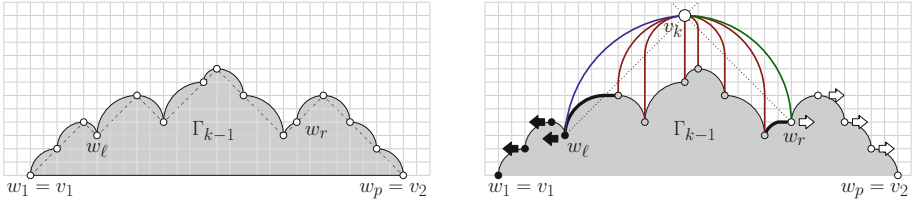


Fig. 9. Illustration of the contour condition (left) and the placement of v_k (right).

i.e., xy -monotone. We achieve our goal at the cost of slightly more edges drawn with complexity 2 or at the cost of increased drawing area (but still polynomial).

Our first approach is a modification of the *shift-method* [19]. Based on a canonical order $\pi = (v_1, \dots, v_n)$ of G , we construct a planar smooth orthogonal 2-drawing Γ of G in the Kandinsky model, as follows. We place v_1, v_2 and v_3 at $(0, 0), (2, 0)$ and $(1, 1)$. Hence, we can draw (v_1, v_2) as a horizontal segment, and each of (v_1, v_3) and (v_2, v_3) as a quarter circular arc. We also color (v_1, v_3) blue and (v_2, v_3) green. For $k = 4, \dots, n$, assume that a smooth orthogonal 2-drawing Γ_{k-1} of the subgraph G_{k-1} of G induced by v_1, \dots, v_{k-1} has been constructed, in which each edge of the outerface C_{k-1} of Γ_{k-1} is drawn as a quarter circular arc, whose endvertices are on a line with slope ± 1 , except for edge (v_1, v_2) , which is drawn as a horizontal segment (called *contour condition* in the shift-method; see Fig. 9). Each of v_1, \dots, v_{k-1} is also associated with a so-called *shift-set*, which for v_1, v_2 and v_3 are singletons containing only themselves.

Let w_1, \dots, w_p be the vertices of C_{k-1} from left to right in Γ_{k-1} , where $w_1 = v_1$ and $w_p = v_2$. Let $(w_\ell, \dots, w_r), 1 \leq \ell < r \leq p$, be the neighbors of v_k from left to right along C_{k-1} in Γ_{k-1} . As in the shift-method, our algorithm first translates each vertex in $\cup_{i=1}^\ell S(w_i)$ one unit to the left and each vertex in $\cup_{i=r}^p S(w_i)$ one unit to the right, where $S(v)$ is the shift-set of $v \in V$. During this translation, $(w_\ell, w_{\ell+1})$ and (w_{r-1}, w_r) acquire a horizontal segment each (see the bold edges of Fig. 9). We place v_k at the intersection of line L_ℓ with slope $+1$ through w_ℓ with line L_r with slope -1 through w_r (dotted in Fig. 9) and we set the shift-set of v_k to $\{v_k\} \cup_{i=\ell+1}^{r-1} S(w_i)$, as in the shift-method. We draw each of (w_ℓ, v_k) and (v_k, w_r) as a quarter circular arc. For $i = \ell + 1, \dots, r - 1$, (w_i, v_k) has a vertical line-segment that starts from w_i and ends either at L_ℓ or L_r and a quarter circle arc from the end of the previous segment to v_k . Hence, the contour condition is satisfied. We color (w_ℓ, v_k) blue, (v_k, w_r) green and the remaining edges of v_k red; see also [16, 33]. Observe that each blue and green edge consists of a quarter circular arc and a horizontal segment (that may have zero length), while a red edge consists of a vertical segment and a quarter circular arc (that may have zero radius). We are now ready to state our first theorem; the analogous of Theorem 6 for the octilinear model is shown in [2].

Theorem 6. *A maximal planar n -vertex graph admits a bi-monotone planar smooth orthogonal 2-drawing in the Kandinsky model, which requires $O(n^2)$ area and can be computed in $O(n)$ time.*

Proof. Bi-monotonicity follows by construction. The time complexity follows from [9]. Planarity is proven by induction. Drawing Γ_3 is planar by construction. Assuming that Γ_{k-1} is planar, we observe that no two edges incident to v_k cross in Γ_k . Also, these edges do not cross edges of Γ_{k-1} . Since the radii of the arcs of the edges incident to vertices that are shifted remain unchanged and since edges incident to vertices in the shift-sets retain their shape, drawing Γ_k is planar. \square

We reduce the number of edges drawn with complexity 2 in two steps. (S.1) We stretch the drawing horizontally (by employing appropriate vertical cuts; see, e.g., [17]) to eliminate the vertical segments of all red edges with a circular arc segment of non-zero radius. (S.2) We stretch the drawing vertically, to guarantee that the edges of a spanning tree (i.e., $n - 1$) are drawn with complexity 1.

For Step 1, we assume that each blue and green edge has a horizontal segment (that may be of zero length). Consider a red edge (u, v) with a vertical segment of length δ and assume w.l.o.g. that u is to the right and above v . If we shift u by δ units to the right, then (u, v) can be drawn as a quarter circular arc. If the shift is by more than δ units, then a horizontal segment is needed. Since all edges incident to u that are drawn below u enter u from its left or from its right side, the shift of u cannot introduce crossings between them.

We eliminate the vertical segments of all red edges with a circular arc segment of non-zero radius, as follows. As long as there exist such edges, we choose the one, call it (u, v) , whose vertical segment has the largest length δ , and assume that u is to the right and above v . We eliminate the vertical segment of (u, v) using a vertical cut L at $x(u) - \varepsilon$, for small $\varepsilon > 0$. Since L crosses several edges, shifting all vertices to the right of L by δ to the right has the following effects. By the choice of (u, v) , the vertical segments of all red edges crossed by L are eliminated; note that this may introduce new horizontal segments. The horizontal segment of each blue and green edge crossed by L is elongated by δ . Both imply that no edge crossings are introduced. Hence, by the termination of our algorithm all edges with vertical segments are of complexity 1.

Step 1 ensures that the x -distance of adjacent vertices is at least as large as their y -distance (unless they are connected by vertical edges). Based on this property, in Step 2 we compute new y -coordinates for the vertices in the sequence of the canonical ordering π , keeping their x -coordinates unchanged. First, we set $y(v_1) = y(v_2) = 0$. For each $k = 3, \dots, n$, we set $y(v_k) = \max_{w \in \{w_\ell, \dots, w_r\}} \{y(w) + \max\{\Delta_x(v_k, w), 1\}\}$, where w_ℓ, \dots, w_r are the neighbors of v_k in Γ_{k-1} , i.e., v_k is placed above w_ℓ, \dots, w_r in Γ_{k-1} , such that one of its edges (the one of the maximum; call it (v_k, w^*)) is drawn with complexity 1; as a quarter circle arc or as a vertical edge depending on whether the x -distance of v_k and w^* is non-zero or not. Since (v_k, w^*) is the edge that must be stretched the most in order to ensure that it is drawn with complexity 1, for all other edges incident to v_k in G_k , the y -distance of their endpoints is at least as large as their corresponding

x -distance. Hence, they are drawn as vertical segments followed by quarter circular arcs (that may have zero radius). We are now ready to state our second theorem.

Theorem 7. *A maximal planar n -vertex graph admits a bi-monotone planar smooth orthogonal 2-drawing with at least $n - 1$ edges with complexity 1 in the Kandinsky model, which requires $O(n^4)$ area and can be computed in $O(n^2)$ time.*

Proof (sketch). For $k = 3, \dots, n$, vertex v_k is incident to an edge drawn with complexity 1 in Step 2. Since (v_1, v_2) is drawn as a horizontal segment, at least $n - 1$ edges have complexity 1. Planarity is proven by induction; the main invariant is that all edges on $C_k \setminus \{(v_1, v_2)\}$ have a quarter circular arc and possibly a vertical segment. Time and area requirements are shown in [2]. \square

5 Conclusions

In this paper, we continued the study on smooth orthogonal and octilinear drawings. Our \mathcal{NP} -hardness proofs are a first step towards settling the complexity of both drawing problems. We conjecture that the former is \mathcal{NP} -hard, even in the case where only the planar embedding is specified by the input. For the latter, it is of interest to know if it remains \mathcal{NP} -hard even for planar graphs of maximum degree 4 or if these graphs allow for a decision algorithm. Our drawing algorithms guarantee bi-monotone 2-drawings with a certain number of complexity-1 edges for maximal planar graphs. Improvements on this number or generalizations to triconnected or simply connected planar graphs are of importance.

Acknowledgements. The authors would like to thank Patrizio Angelini and Martin Gronemann for useful discussions.

References

1. Alam, M.J., Bekos, M.A., Kaufmann, M., Kindermann, P., Kobourov, S.G., Wolff, A.: Smooth orthogonal drawings of planar graphs. In: Pardo, A., Viola, A. (eds.) LATIN 2014. LNCS, vol. 8392, pp. 144–155. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54423-1_13
2. Bekos, M.A., Förster, H., Kaufmann, M.: On smooth orthogonal and octilinear drawings: Relations, complexity and kandinsky drawings. CoRR 1708.09197 (2017)
3. Bekos, M.A., Gronemann, M., Kaufmann, M., Krug, R.: Planar octilinear drawings with one bend per edge. *J. Graph Algorithms Appl.* **19**(2), 657–680 (2015)
4. Bekos, M.A., Gronemann, M., Pupyrev, S., Raftopoulou, C.N.: Perfect smooth orthogonal drawings. In: Bourbakis, N.G., Tsihrintzis, G.A., Virvou, M. (eds.) IISA, pp. 76–81. IEEE (2014)
5. Bekos, M.A., Kaufmann, M., Kobourov, S.G., Symvonis, A.: Smooth orthogonal layouts. *J. Graph Algorithms Appl.* **17**(5), 575–595 (2013)
6. Bekos, M.A., Kaufmann, M., Krug, R.: On the total number of bends for planar octilinear drawings. *J. Graph Algorithms Appl.* **21**(4), 709–730 (2017)

7. Bertolazzi, P., Di Battista, G., Didimo, W.: Computing orthogonal drawings with the minimum number of bends. *IEEE Trans. Comput.* **49**(8), 826–840 (2000)
8. Biedl, T.C., Kant, G.: A better heuristic for orthogonal graph drawings. *Comput. Geom.* **9**(3), 159–180 (1998)
9. Chrobak, M., Payne, T.H.: A linear-time algorithm for drawing a planar graph on a grid. *Inf. Process. Lett.* **54**(4), 241–246 (1995)
10. Di Battista, G., Didimo, W., Patrignani, M., Pizzonia, M.: Orthogonal and quasi-upward drawings with vertices of prescribed size. In: Kratochvíl, J. (ed.) *GD 1999*. LNCS, vol. 1731, pp. 297–310. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-46648-7_31
11. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, Upper Saddle River (1999)
12. Didimo, W., Liotta, G., Patrignani, M.: On the complexity of HV-rectilinear planarity testing. In: Duncan, C., Symvonis, A. (eds.) *GD 2014*. LNCS, vol. 8871, pp. 343–354. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45803-7_29
13. Duncan, C.A., Eppstein, D., Goodrich, M.T., Kobourov, S.G., Nöllenburg, M.: Lombardi drawings of graphs. *J. Graph Algorithms Appl.* **16**(1), 85–108 (2012)
14. Durocher, S., Felsner, S., Mehrabi, S., Mondal, D.: Drawing HV-restricted planar graphs. In: Pardo, A., Viola, A. (eds.) *LATIN 2014*. LNCS, vol. 8392, pp. 156–167. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54423-1_14
15. Eppstein, D.: Planar lombardi drawings for subcubic graphs. In: Didimo, W., Patrignani, M. (eds.) *GD 2012*. LNCS, vol. 7704, pp. 126–137. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36763-2_12
16. Felsner, S.: *Geometric Graphs and Arrangements*. Advanced Lectures in Mathematics. Vieweg, Wiesbaden (2004). <https://doi.org/10.1007/978-3-322-80303-0>
17. Fößmeier, U., Heß, C., Kaufmann, M.: On improving orthogonal drawings: The 4M-algorithm. In: Whitesides, S.H. (ed.) *GD 1998*. LNCS, vol. 1547, pp. 125–137. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-37623-2_10
18. Fößmeier, U., Kaufmann, M.: Drawing high degree graphs with low bend numbers. In: Brandenburg, F.J. (ed.) *GD 1995*. LNCS, vol. 1027, pp. 254–266. Springer, Heidelberg (1996). <https://doi.org/10.1007/BFb0021809>
19. de Fraysseix, H., Pach, J., Pollack, R.: How to draw a planar graph on a grid. *Combinatorica* **10**(1), 41–51 (1990)
20. Fulek, R., Pelsmajer, M.J., Schaefer, M., Štefankovič, D.: Hanani-Tutte and monotone drawings. In: Kolman, P., Kratochvíl, J. (eds.) *WG 2011*. LNCS, vol. 6986, pp. 283–294. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25870-1_26
21. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York (1979)
22. Garg, A., Tamassia, R.: On the computational complexity of upward and rectilinear planarity testing. *SIAM J. Comput.* **31**(2), 601–625 (2001)
23. Di Giacomo, E., Liotta, G., Montecchiani, F.: The planar slope number of subcubic graphs. In: Pardo, A., Viola, A. (eds.) *LATIN 2014*. LNCS, vol. 8392, pp. 132–143. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54423-1_12
24. Harary, F.: *Graph Theory*. Addison-Wesley, MA (1991)
25. Hong, S., Merrick, D., do Nascimento, H.A.D.: Automatic visualisation of metro maps. *J. Vis. Lang. Comput.* **17**(3), 203–224 (2006)
26. Kant, G.: Drawing planar graphs using the canonical ordering. *Algorithmica* **16**(1), 4–32 (1996)

27. Kaufmann, M., Wagner, D. (eds.): Drawing Graphs. LNCS, vol. 2025. Springer, Heidelberg (2001). <https://doi.org/10.1007/3-540-44969-8>
28. Keszegh, B., Pach, J., Pálvölgyi, D.: Drawing planar graphs of bounded degree with few slopes. *SIAM J. Discrete Math.* **27**(2), 1171–1183 (2013)
29. Liu, Y., Morgana, A., Simeone, B.: A linear algorithm for 2-bend embeddings of planar graphs in the two-dimensional grid. *Discrete Appl. Math.* **81**(1–3), 69–91 (1998)
30. Mañuch, J., Patterson, M., Poon, S.-H., Thachuk, C.: Complexity of finding non-planar rectilinear drawings of graphs. In: Brandes, U., Cornelsen, S. (eds.) GD 2010. LNCS, vol. 6502, pp. 305–316. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-18469-7_28
31. Nöllenburg, M.: Automated Drawing of Metro Maps. Master’s thesis, Fakultät für Informatik, Universität Karlsruhe (TH), August 2005
32. Nöllenburg, M., Wolff, A.: Drawing and labeling high-quality metro maps by mixed-integer programming. *IEEE Trans. Vis. Comput. Graph.* **17**(5), 626–641 (2011)
33. Schnyder, W.: Embedding planar graphs on the grid. In: Johnson, D.S. (ed.) SODA, pp. 138–148. SIAM (1990)
34. Stott, J.M., Rodgers, P., Martinez-Ovando, J.C., Walker, S.G.: Automatic metro map layout using multicriteria optimization. *IEEE Trans. Vis. Comput. Graph.* **17**(1), 101–114 (2011)
35. Tamassia, R.: On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.* **16**(3), 421–444 (1987)

EPG-representations with Small Grid-Size

Therese Biedl¹(✉), Martin Derka¹, Vida Dujmović², and Pat Morin³

¹ Cheriton School of Computer Science,
University of Waterloo, Waterloo, ON, Canada
{biedl, mderka}@uwaterloo.ca

² School of Computer Science and Electrical Engineering,
University of Ottawa, Ottawa, ON, Canada
vida.dujmovic@uottawa.ca

³ School of Computer Science, Carleton University, Ottawa, ON, Canada
morin@cs.carleton.ca

Abstract. In an EPG-representation of a graph G , each vertex is represented by a path in the rectangular grid, and (v, w) is an edge in G if and only if the paths representing v and w share a grid-edge. Requiring paths representing edges to be x-monotone or, even stronger, both x- and y-monotone gives rise to three natural variants of EPG-representations, one where edges have no monotonicity requirements and two with the aforementioned monotonicity requirements. The focus of this paper is understanding how small a grid can be achieved for such EPG-representations with respect to various graph parameters.

We show that there are m -edge graphs that require a grid of area $\Omega(m)$ in any variant of EPG-representations. Similarly there are pathwidth- k graphs that require height $\Omega(k)$ and area $\Omega(kn)$ in any variant of EPG-representations. We prove a matching upper bound of $O(kn)$ area for all pathwidth- k graphs in the strongest model, the one where edges are required to be both x- and y-monotone. Thus in this strongest model, the result implies, for example, $O(n)$, $O(n \log n)$ and $O(n^{3/2})$ area bounds for bounded pathwidth graphs, bounded treewidth graphs and all classes of graphs that exclude a fixed minor, respectively. For the model with no restrictions on the monotonicity of the edges, stronger results can be achieved for some graph classes, for example an $O(n)$ area bound for bounded treewidth graphs and $O(n \log^2 n)$ bound for graphs of bounded genus.

1 Introduction

The $w \times h$ -grid (or *grid of width w and height h*) consists of all *grid-points* (i, j) that have integer coordinates $1 \leq i \leq w$ and $1 \leq j \leq h$, and all *grid-edges* that connect grid-points of distance 1. An *EPG-representation* of a graph G consists

Work done during the 5th Workshop on Graphs and Geometry, Bellairs Research Institute. The authors would like to thank the other participants, and especially Günter Rote, for helpful input. Research of TB, VD and PM supported by NSERC. Research of MD supported by an NSERC Vanier scholarship.

of an assignment of a *vertex-path*, $path(v)$, to every vertex v in G such that $path(v)$ is a path in a grid, and (v, w) is an edge of G if and only if $path(v)$ and $path(w)$ have a grid-edge in common.

Since their initial introduction by Golumbic et al. [11], a number of papers concerning EPG-representations of graphs have been published. It is easy to see that *every* graph has an EPG-representation [11]. Later papers asked what graph classes can be represented if the number of bends in the vertex-paths is restricted (see e.g. [3, 4, 9, 13]) or gave approximation algorithms for graphs with an EPG-representation with few bends (see e.g. [8, 17]).

The main objective of this paper is to find EPG-representations such that the size of the underlying grid is small (rather than the number of bends in vertex-paths). As done by Golumbic et al. [11], we wonder whether additionally we can achieve monotonicity of vertex-paths. We say that $path(v)$ is *x-monotone* if any vertical line that intersects $path(v)$ intersects it in a single interval. It is *xy-monotone* if it is *x-monotone* and additionally any horizontal line that intersects $path(v)$ intersects it in a single interval. Finally, it is *xy⁺-monotone* if it is monotonically increasing, i.e., it is *xy-monotone* and the left endpoint is not above the right endpoint. An *x-monotone EPG-representation* is an EPG-representation where every vertex-path is *x-monotone*, and similarly an *xy⁺-monotone EPG-representation* is an EPG-representation where every vertex-path is *xy⁺-monotone*.

It is easy to see that every n -vertex graph has an EPG-representation in an $O(n) \times O(n)$ -grid, i.e., with quadratic area. This is best possible for some graphs. In Sect. 4, we study lower bounds and show that there are m -edge graphs that require a grid of area $\Omega(m)$ in any EPG-representation and that there are pathwidth- k graphs that require height $\Omega(k)$ and area $\Omega(kn)$ in any EPG-representation.

Biedl and Stern [4] showed that pathwidth- k graphs have an EPG-representation of height k and width $O(n)$, thus area $O(kn)$. In Sect. 5, we prove a strengthening of that result. In particular, we show that every pathwidth- k graph has an *xy⁺-monotone* EPG-representation of height $O(k)$ and width $O(n)$ thus matching the lower bound in this strongest of the models. This result implies, for example, $O(n)$, $O(n \log n)$ and $O(n^{3/2})$ area bounds for *xy⁺-monotone* EPG-representations of bounded pathwidth graphs, bounded treewidth graphs and all classes of graphs that exclude a minor, respectively. In fact, the result implies that all hereditary graph classes with $o(n)$ -size balanced separators have $o(n^2)$ area *xy⁺-monotone* EPG-representations.

If the monotonicity requirement is dropped, better area bounds are possible for some graph classes. For example, in Sect. 6, we prove that graphs of bounded treewidth have $O(n)$ area EPG-representations and that graphs of bounded genus (thus planar graphs too) have $O(n \log^2 n)$ EPG-representations.

2 Preliminaries

Throughout this paper, $G = (V, E)$ denotes a graph with n vertices and m edges. We refer, e.g., to [6] for all standard notations for graphs. The *pathwidth* $pw(G)$

of a graph G is a well-known graph parameter. Among the many equivalent definitions, we use here the following: $pw(G)$ is the smallest k such that there exists a super-graph H of G that is a $(k+1)$ -colourable interval graph. Here, an *interval graph* is a graph that has an *interval representation*, i.e., an assignment of a (1-dimensional) interval to each vertex v such that there exists an edge if and only if the two intervals share a point. We may without loss of generality assume that the intervals begin and end at distinct x -coordinates in $\{1, \dots, 2n\}$, and will do so for all interval representations used in this paper. We use $I(v) = [\ell(v), r(v)]$ for the interval representing vertex v . It is well-known that an interval graph is k -colourable if and only if its maximum clique size is k .

Contracting an edge (v, w) of a graph G means deleting both v and w , inserting a new vertex x , and making x adjacent to all vertices in $G - \{v, w\}$ that were adjacent to v or w . A graph H is called a *minor* of a graph G if H can be obtained from G by deleting some vertices and edges of G and then contracting some edges of G . It is known that $pw(H) \leq pw(G)$ for any minor H of G .

3 From Proper VPG to EPG

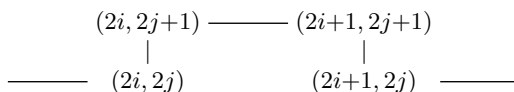
A *VPG-representation* of a graph G consists of an assignment of vertex-paths in the grid to vertices of G such that (v, w) is an edge of G if and only if $path(v)$ and $path(w)$ have a grid-point in common. Many previous EPG-representation constructions (see e.g. [11]) were obtained by starting with a VPG-representation and transforming it into an EPG-representation by adding a “bump” whenever two paths cross. The lemmas below formalize this idea, and also study how this transformation affects the grid-size and whether monotonicity is preserved.

We give the transformations only for *proper* VPG-representations, which satisfy the following: (a) Any grid-edge is used by at most one vertex-path. (b) If a grid-point p belongs to $path(v)$ and $path(w)$, then one of the vertex-paths includes the rightward edge at p and the other includes the upward edge at p ¹.

Lemma 1. *Let G be a graph that has a proper VPG-representation R_V in a $w \times h$ -grid. Then any subgraph G' of G has an EPG-representation R_E in a $2w \times 2h$ -grid. Furthermore, if R_V is x -monotone then R_E is x -monotone.*

Proof. Double the resolution of the grid by inserting a new grid-line after each existing one. For each edge (v, w) of G' , consider the two paths $path(v)$ and $path(w)$ that represent v and w in R_V . Since (v, w) was an edge of G , the vertex-paths share a grid-point (i, j) in R_V , which corresponds to point $(2i, 2j)$ in R_E .

Since R_V is proper, we may assume (after possible renaming) that $path(v)$ uses the rightward edge at $(2i, 2j)$, and $path(w)$ uses the upward edge at $(2i, 2j)$. Re-route $path(v)$ by adding a “bump”



¹ The transformation could be done with a larger factor of increase if (b) is violated, but restriction (a) is vital.

in the first quadrant of $(2i, 2j)$. See also Fig. 1(a) and (b). Note that $path(w)$ is unchanged in the vicinity of $(2i, 2j)$, and the bump added to $path(v)$ is x -monotone. So if R_V is x -monotone then so are the resulting vertex-paths.

Since R_V is proper, no other vertex-path used (i, j) in R_V , and therefore no other vertex-path in R_E can use any grid-edge of this bump. Therefore no new adjacencies are created, and R_E is indeed an EPG-representation of G' . \square

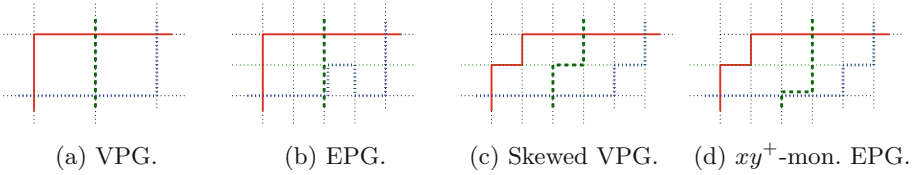


Fig. 1. Transforming a proper VPG-representation. We only show the transformation for the edge from blue (dotted) to green (dashed) vertex. (Color figure online)

We now give a second construction, which is similar in spirit, but re-routes differently in order to preserve xy^+ -monotonicity.

Lemma 2. *Let G be a graph that has a proper VPG-representation R_V in a $w \times h$ -grid with xy^+ -monotone vertex-paths. Then any subgraph G' of G has an xy^+ -monotone EPG-representation R_E in a $(2w + h) \times 2h$ -grid*

Proof. We do two transformations; the first results in a proper VPG-representation R'_V that has some special properties such that it can then be transformed into an EPG-representation.

The first transformation is essentially a skew. Map each grid-point (i, j) of R_V into the corresponding point $(2i + j, 2j)$. Any horizontal grid-edge used by a vertex-path is mapped to the corresponding horizontal grid-edge, i.e., we map a horizontal grid-edge $(i, j) - (i+1, j)$ of R_V into the length-2 horizontal segment $(2i + j, 2j) - (2(i + 1) + j, 2j)$ that connects the corresponding points. Every vertical grid-edge $(i, j) - (i, j+1)$ is mapped into the zig-zag path

$$\begin{array}{ccc}
 & & (2i+(j+1), 2(j+1)) \\
 & & | \\
 (2i+j, 2j+1) & \text{---} & (2i+j+1, 2j+1) \\
 | & & | \\
 (2i+j, 2j) & &
 \end{array}$$

that connects the corresponding points. See also Fig. 1(a) and (c). It is easy to verify that this is again a proper VPG-representation of exactly the same graph, and vertex-paths are again xy^+ -monotone.

Now view R'_V as an EPG-representation. Since R'_V is proper, currently no edge is represented. We now modify R'_V such that intersections are created if and only if an edge exists. Consider some edge (v, w) of G' . Since

it is an edge of G , there must exist a point (i, j) in R_V where $path(v)$ and $path(w)$ meet. Since R_V is proper, we may assume (after possible renaming) that $path(v)$ uses the rightward edge at (i, j) while $path(w)$ uses the upward edge at (i, j) . Consider the corresponding point $(2i + j, 2j)$ in R'_V , and observe that $path'(v)$ and $path'(w)$ (the vertex-paths in R'_V) use its incident rightward and upward edges, respectively. Moreover, $path'(w)$ uses the “zig-zag” $(2i + j, 2j) - (2i + j, 2j + 1) - (2i + j + 1, 2j + 1)$. We can now re-route the vertex-path of w to use instead $(2i + j, 2j) - (2i + j + 1, 2j) - (2i + j + 1, 2j + 1)$, i.e., to share the horizontal edge with $path'(w)$ and then go vertically. See Fig. 1(d). Thus the two paths now share a grid-edge. Since no other vertex-paths used (i, j) in R_V , this re-routing does not affect any other intersections and overlaps. So we obtain an EPG-representation of G , and one easily verifies that it is xy^+ -monotone. \square

Theorem 1. *Every graph G with n vertices has an xy^+ -monotone EPG-representation in a $3n \times 2n$ -grid.*

Proof. It is very easy to create a proper VPG-representation of the complete graph K_n in an $n \times n$ -grid, using a Γ -shape (hence an xy^+ -monotone vertex-path). Namely, place the corner of the Γ of vertex i at $(i - 1, i)$ and extending the two arms to $y = 1$ and $x = n$. For vertex 1, the grid-edge $(0, 1) - (1, 1)$ is not needed and can be omitted to save a column. See Fig. 2. Since G is a subgraph of K_n , the result then follows by Lemma 2. \square

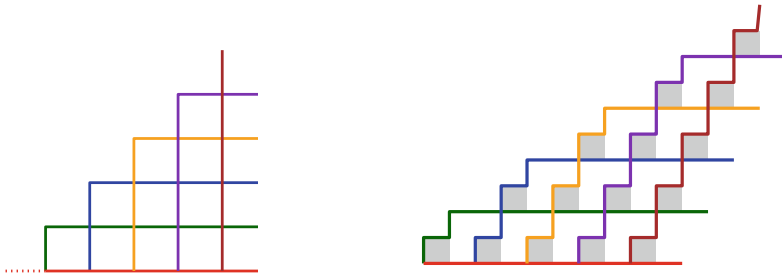


Fig. 2. A VPG-representation of K_n , and an EPG-representation for any graph. In gray areas vertex-paths may get re-routed to create shared grid-edges.

Contrasting this with existing results, it was already known that any graph has an EPG-representation [11], but our construction additionally imposes xy^+ -monotonicity, and our grid-size is $O(n^2)$, rather than $O(nm)$.

4 Lower Bounds

We now turn to lower bounds. These hold for arbitrary EPG-representations; we make no use of monotonicity.

Theorem 2. *Let G be a triangle-free graph with m edges. Then any EPG-representation of G uses at least m grid-edges (hence a grid of area $\Omega(m)$).*

Proof. If G has no triangle, then the maximal clique-size is 2. Hence no grid-edge can belong to three or more vertex-paths. Consequently, for every edge (v, w) we must have at least one grid-edge (the one that is common to $path(v)$ and $path(w)$). No grid-edge belongs to three vertex-paths, and so there must be at least m grid-edges. \square

A consequence of Theorem 2 is that $K_{n,n}$ requires $\Omega(n^2)$ area in any EPG-representation. Later, we relate pathwidth to EPG-representations. For now, we note that $K_{n-k,k}$ is an n -vertex triangle-free graph with pathwidth k and $\Theta(kn)$ edges. Together with Theorem 2, this implies:

Corollary 1. *For every $k \geq 1$ and every $n \geq 2k$, there exists an n -vertex pathwidth- k graph G for which any EPG-representation of G uses $\Omega(kn)$ grid-edges (hence a grid of area $\Omega(kn)$).*

One wonders whether there are graphs that have only a linear number of edges and still require a big, even quadratic, area. The following lower bound, also based on pathwidth, allows us to answer this question in the affirmative.

Theorem 3. *Let G be a graph that has an EPG-representation in a grid with h rows and for which any grid-edge is used by at most c vertex-paths. Then $pw(G) \leq c(3h - 1) - 1$.*

Proof. For every vertex v , define $I(v)$ to be the x -projection of $path(v)$. This is an interval since $path(v)$ is connected. Define H to be the interval graph of these intervals. If (v, w) is an edge, then $path(v)$ and $path(w)$ share a grid-edge, and hence the intervals $I(v)$ and $I(w)$ share at least one point. So G is a subgraph of H . We claim that H has clique-size $\omega(H) \leq 6h - 2$; this implies the result.

Fix an arbitrary maximal clique D in H . It is well-known (see e.g. [10]) that, in the projected interval-representation, there exists a vertex v such that D corresponds to those vertices whose intervals intersect the left endpoint $\ell(v)$. Hence for any vertex w in D , at least one grid-edge of $path(w)$ is incident to a grid-point with x -coordinate $\ell(v)$. There are only $3h - 1$ such grid-edges ($2h$ horizontal ones and $h - 1$ vertical ones), and each of them can belong to at most c vertex-paths. Hence $|D| \leq c(3h - 1)$, which proves the claim. \square

In particular, if G is triangle-free then no three vertex-paths can share a grid-edge. Applying the theorem with $c = 2$ for such graphs we get:

Corollary 2. *Any triangle-free graph with pathwidth k requires an $\Omega(k) \times \Omega(k)$ -grid and thus $\Omega(k^2)$ area in any EPG-representation.*

So all that remains to do for a better lower bound is to find a graph that has few edges yet high pathwidth. For this, we use *expander-graphs*, which are graphs such that for any vertex-set S the ratio between the boundary of S (the number of vertices in S with neighbors in $V - S$) and $|S|$ is bounded from below.

Theorem 4. *There are n -vertex graphs with $O(n)$ edges for which any EPG-representation requires $\Omega(n^2)$ area.*

Proof. It is known that expander-graphs of maximum degree 3 exist (see e.g. [15]) Let G be one such graph. It hence has $O(n)$ edges. Since G is an expander, it has pathwidth $\Omega(n)$ (see e.g. [12]). Subdivide all edges of G to obtain a bipartite graph G' that has $O(n)$ vertices and edges. This operation cannot decrease the pathwidth since G is a minor of G' . So $pw(G') \in \Omega(n)$. Since G' is triangle-free, any EPG-representation of G' must have height $\Omega(n)$, and a symmetric argument shows that it must have width $\Omega(n)$. \square

5 Upper Bounds on xy^+ -monotone EPG Representations

Corollaries 1 and 2 imply that the best upper-bounds for EPG-representations in terms of pathwidth have height $\Omega(k)$ and area $\Omega(kn)$. Naturally, one wonders whether this bound can be matched. As noted in the introduction, Biedl and Stern showed that any graph with pathwidth k has an EPG-representation of height k and area $O(kn)$ [4]. We now use a completely different approach to strengthen their result and obtain xy^+ -monotone EPG-representations of pathwidth k graphs with optimal height $O(k)$ and optimal area $O(kn)$.

Theorem 5. *Every graph G of pathwidth k has an xy^+ -monotone EPG-representation of height $8k + O(1)$ and width $O(n)$, thus with $O(kn)$ area.*

Proof. Recall that G is a subgraph of a $(k+1)$ -colourable interval graph H . By Lemma 2, it suffices to show the following:

Lemma 3. *Let H be a $(k+1)$ -colourable interval graph with interval representation $\{I(v) = [\ell(v), r(v)] : v \in V\}$. There exists a proper VPG-representation with xy^+ -monotone vertex-paths of a supergraph of H such that*

1. *all vertex-paths are contained within the $[2, 2n + 1] \times [-2k - 2, 2k + 1]$ -grid (more precisely, the x -range is $[2 \min_{v \in V} \ell(v), 1 + 2 \max_{v \in V} r(v)]$);*
2. *$path(v)$ contains a horizontal segment whose x -range is $[2\ell(v), 2r(v)]$ and whose y -coordinate is negative; and*
3. *some vertical segment $path(v)$ includes the segment $\{2r(v)\} \times [-1, 1]$.*

We prove the lemma by induction on k . We may assume that H is connected, for if it is not, then obtain representations of each connected component separately and combine them. The x -ranges of intervals of each component are disjoint (else there would be an edge), and so the representations of the components do not overlap by (1).

The claim is straightforward for $k = 0$: Since H is connected and 1-colourable, it has only one vertex v . Set $path(v)$ to use the two segments $[2\ell(v), 2r(v)] \times \{-1\}$ and $\{2r(v)\} \times [-1, 1]$. All claims hold.

Now assume that $k \geq 1$. We find a path P of “farthest-reaching” intervals as follows. Set $a_1 := \operatorname{argmin}_{v \in V} \ell(v)$, i.e., a_1 is the interval that starts leftmost.

Assume a_i has been defined for some $i \geq 1$. Let \mathcal{A}_i be the set of all vertices v with $\ell(a_i) < \ell(v) < r(a_i) < r(v)$. If \mathcal{A}_i is empty then stop the process; we have reached the last vertex of P . Else, set $a_{i+1} := \operatorname{argmax}_{v \in \mathcal{A}_i} r(v)$ to be the vertex in \mathcal{A}_i whose interval goes farthest to the right, and repeat. See also Fig. 3. Let $P = a_1, a_2, \dots, a_p$ be the path that we obtained (this is indeed a path since $I(a_i)$ intersects $I(a_{i+1})$ by definition).

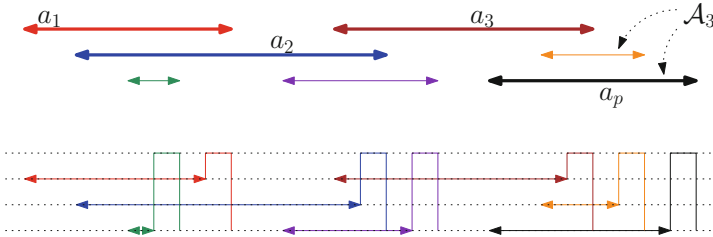


Fig. 3. An interval graph (bold intervals denote the path P chosen in Theorem 5), and its proper VPG-representation with x -monotone vertex-paths.

Claim. P is an induced path.

Proof. It suffices to show that $r(a_i) < \ell(a_{i+2})$ for all $1 \leq i \leq p - 2$. Assume for contradiction that $\ell(a_{i+2}) < r(a_i)$ for some $1 \leq i \leq p - 2$. We show that this contradicts the choice of P as the vertices that go farthest right. Namely, let $j \leq i$ be the smallest index such that $\ell(a_{i+2}) < r(a_j)$. If $j > 1$ then $\ell(a_{i+2}) > r(a_{j-1}) > \ell(a_j)$ by definition of j and \mathcal{A}_{j-1} . If $j = 1$ then $\ell(a_{i+2}) \geq \min_{v \in V} \ell(v) = \ell(a_1) = \ell(a_j)$, and the inequality is strict since $i+2 \neq 1$. Thus in both cases $\ell(a_{i+2}) > \ell(a_j)$. Therefore $\ell(a_j) < \ell(a_{i+2}) < r(a_j) \leq r(a_{i+1}) < r(a_{i+2})$, which implies $a_{i+2} \in \mathcal{A}_j$. By $r(a_{j+1}) \leq r(a_{i+1}) < r(a_{i+2})$ this contradicts the choice of a_{j+1} as $\operatorname{argmax}_{v \in \mathcal{A}_j} r(v)$. \square

By definition a_1 is the leftmost interval, i.e., $\ell(a_1) = \min_{v \in V} \ell(v)$. We claim that a_p is the rightmost interval, i.e., $r(a_p) = \max_{v \in V} r(v)$. Assume for contradiction that some vertex v has an interval that ends farther right. By connectivity we can choose v so that it intersects $I(a_p)$, thus $\ell(v) < r(a_p) < r(v)$. Let $j \leq p$ be maximal such that $\ell(v) < r(a_j)$. Similarly, as in the claim, one argues that $v \in \mathcal{A}_j$, and therefore v , rather than a_{j+1} , should have been added to path P .

We are now ready for the construction. Define $H' := H - P$. Since the intervals of P cover the entire range $[\min_{v \in V} \ell(v), \max_{v \in V} r(v)]$, any maximal clique of H contains a vertex of P . Therefore the maximum clique-size of H' satisfies $\omega(H') \leq \omega(H) - 1$, which implies (for an interval-graph) that $\chi(H') \leq \chi(H) - 1$, hence H' is k -colourable. Apply induction to H' (with the induced interval representation) and let Γ' be the resulting VPG-representation.

Since I' uses only orthogonal vertex-paths, we can insert two rows each above and below the x -axis by moving all other bends up/down appropriately. Now set $path(a_i)$ to be

$$\begin{array}{ccc}
 & (2r(a_i), -Y) & \text{---} & (2r(a_{i+1}) + 1, Y) \\
 & | & & | \\
 (2\ell(a_i), -Y) & \text{---} & (2r(a_i), -Y) & \\
 | & & & \\
 (2\ell(a_i), -2k - 2) & & &
 \end{array}$$

where $Y = 1$ if i is odd and $Y = 2$ if i is even. We omit the rightmost horizontal segment for $i = p$ (because a_{p+1} is undefined). See also Fig. 4.

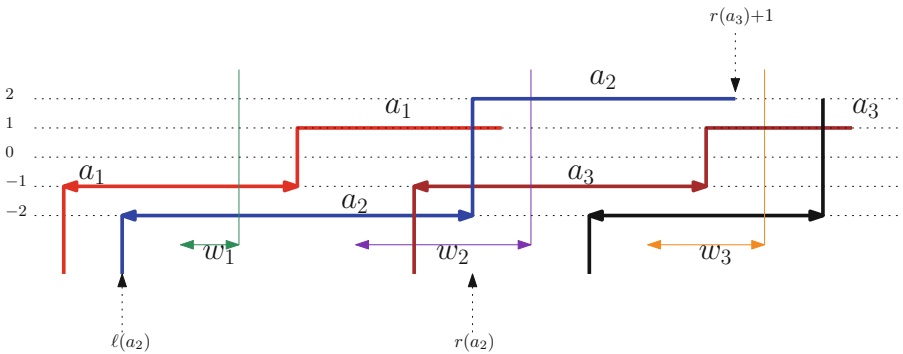


Fig. 4. Representation with xy -monotone vertex-paths.

Note that these vertex-paths satisfy conditions (2) and (3). Also note that for any $1 \leq i < p$, the vertex-paths of a_i and a_{i+1} intersect, namely at $(2r(a_{i+1}), 1)$ if i is odd and at $(2\ell(a_{i+1}), -2)$ and $(2r(a_i), -1)$ if i is even. It remains to show that for any edge (w, a_i) (for some $1 \leq i \leq p$ and $w \notin P$) the vertex-paths intersect. Here we have three cases (Fig. 4 illustrates the ℓ th case for edge $(w_\ell, a_{\ell+1})$):

1. If $r(w) < r(a_i)$, then $\ell(a_i) < r(w)$, else the intervals would not intersect. By (3), and since we inserted new rows around the x -axis, we know that $path(w)$ contains the vertical segment $2r(w) \times [-3, 3]$. Therefore $path(a_i)$ intersects this segment at $(2r(w), -Y)$ where $Y \in \{1, 2\}$.
2. If $\ell(w) < \ell(a_i)$, then $\ell(a_i) < r(w)$, else the intervals would not intersect. By (2), and since we inserted new rows around the x -axis, we know that $path(w)$ has a horizontal segment $[2\ell(w), 2r(w)] \times -Y$ for some $Y \geq 3$. Therefore $path(a_i)$ intersects this segment at $(2\ell(a_i), -Y)$.
3. Finally assume that $\ell(a_i) < \ell(w)$ and $r(a_i) < r(w)$. We must have $\ell(w) < r(a_i)$, else the intervals would not intersect. Therefore $w \in \mathcal{A}_i$. By choice of a_{i+1} we have $r(w) \leq \max_{v \in \mathcal{A}_i} r(v) = r(a_{i+1})$. By (3), and since we inserted new rows around the x -axis, we know that $path(w)$ contains the vertical

segment $2r(w) \times [-3, 3]$. By $r(w) \leq r(a_{i+1})$, therefore $path(a_i)$ intersects this segment at $(2r(w), Y)$ where $Y \in \{1, 2\}$.

Hence all edges of H are represented by intersection of vertex-paths and as one easily verifies, these are proper intersections. This finishes the induction and proves the theorem. \square

We can use Theorem 5 to obtain small EPG-representations for other graph classes. Graphs of bounded treewidth have pathwidth at most $O(\log n)$ [5]. Graphs excluding a fixed minor have treewidth $O(\sqrt{n})$ [2]. A graph class has treewidth $O(n^\epsilon)$ if it is *hereditary* (subgraphs also belong to the class) and has *balanced separators* (for any weight-function on the vertices there exists a small set S such that removing S leaves only components with at most half the weight) for which the *size* (the cardinality of S) is at most $O(n^\epsilon)$, for some fixed $\epsilon \in (0, 1)$ [7]. It is well known that hereditary graph classes with treewidth $O(n^\epsilon)$, for some fixed $\epsilon \in (0, 1)$, have pathwidth $O(n^\epsilon)$ (see [5] for example), so graphs excluding a fixed minor have pathwidth $O(\sqrt{n})$ and graphs with $O(n^\epsilon)$ -size balanced separators have pathwidth $O(n^\epsilon)$. This implies:

- Graphs of bounded treewidth have xy^+ -monotone EPG-representations in an $O(\log n) \times O(n)$ -grid.
- Graphs excluding a fixed minor have xy^+ -monotone EPG-representations in an $O(\sqrt{n}) \times O(n)$ -grid.
- Hereditary classes of graphs with $O(n^\epsilon)$ -sized balanced separators for some $\epsilon \in (0, 1]$ have xy^+ -monotone EPG-representations in an $O(n^\epsilon) \times O(n)$ -grid.

The $O(n)$ area result for bounded pathwidth graphs is tight by Theorem 2. Naturally, one wonders if the other three results in above are tight. The $O(\sqrt{n}) \times O(n)$ -grid bound applies, for example, to all planar graphs and more generally all bounded genus graphs. Some planar graphs with n vertices have pathwidth $\Omega(\sqrt{n})$ (the $\sqrt{n} \times \sqrt{n}$ -grid is one example), so the height cannot be improved. But can the width or the area be improved? This turns out to be true for some graph classes if the monotonicity condition is dropped. In the next section, we show these improved bounds via a detour into orthogonal drawings. Some of these results are tight.

6 EPG-representations via Orthogonal Drawings

In this section, we study another method of obtaining EPG-representations, which gives (for some graph classes) even smaller EPG-representations. Define a *4-graph* to be a graph where all vertices have degree at most 4. An *orthogonal drawing* of a 4-graph is an assignment of grid-points to vertices and grid-paths to edges such that the path of each edge connects the grid points of its end-vertices. Edges are allowed to intersect, but any such intersection point must be a true intersection, i.e., one edge uses only horizontal grid-edges while the other uses only vertical grid-edges at the intersection point.

Lemma 4. *Let G be a 4-graph that has an orthogonal drawing in a $w \times h$ -grid. Then any minor of G has an EPG-representation in a $2w \times 2h$ -grid.*

Proof. First delete from the orthogonal drawing all edges of G that are not needed for the minor H ; this cannot increase the grid-size. So we may assume that H is obtained from G via edge contractions only.

We first explain how to obtain an EPG-representation of G . Double the grid by inserting a new row/column after each existing one. Every grid-point that belonged to a vertex v hence now corresponds to 4 grid-points that form a unit square; denote this by \square_v . Duplicate all segments of grid-paths for edges in the adjacent new grid-line, and extend/shorten suitably so that the copies again form grid-paths, connecting the squares of their end. Thus for each edge (v, w) we now have two grid-paths $P_{v,w}^1$ and $P_{v,w}^2$ from \square_v to \square_w .

We now define $path(v)$ (which will be a closed path) by tracing the edges of the orthogonal drawing suitably. To describe this in more detail, first arbitrarily direct the edges of G . Initially, $path(v)$ is simply the boundary of \square_v . Now consider each edge (v, w) incident to v . If it is directed $v \rightarrow w$, then remove from $path(v)$ the grid-edge along \square_v that connects the two ends of $P_{v,w}^1$ and $P_{v,w}^2$, add these two grid-paths, and add the grid-edge e' along \square_w that connects these two paths. Note that e' also belongs to $path(w)$, so with this $path(v)$ and $path(w)$ share a grid-edge and we obtain the desired EPG-representation of G . See Fig. 5.

It remains to argue that this can be turned into an EPG-representation of a graph H obtained from G via edge contractions. Suppose we want to contract edge (v, w) . The two grid-paths $path(v)$ and $path(w)$ share a grid-edge e that belongs to no other vertex-path. Delete e from both paths, and let the path of the contraction-vertex be the union of the two resulting open paths, which is again a closed path. Thus we obtain an EPG-representation of H where all vertex-paths are closed paths.

If desired, we can turn this into an EPG-representation with open paths by deleting for every $v \in V$ one grid-edge from $path(v)$ that is not shared with any other vertex-path. If $\deg(v) \leq 3$, then a suitable edge is the grid-edge of \square_v on the side where no edge attaches. If v has an outgoing edge $v \rightarrow w$, then a suitable edge is any grid-edge of $P_{v,w}^1$. We can achieve that one of these always holds as follows: If all vertex degrees of G are 4, then direct G by walking along an Eulerian cycle; then all vertices have outgoing edges. If some vertex v has degree 3 or less, then find a spanning tree T of G , root it at v , direct all tree-edges towards the root and all other arbitrarily. Either way, this direction satisfies that any vertex of degree 4 has at least one outgoing edge and we can delete an edge of each $path(v)$ such that all vertex-paths are open paths. \square

We note that a somewhat similar transformation from orthogonal drawings was used recently to create pixel-representations [1], but in contrast to their result we do not need the orthogonal drawings to be planar. We use this lemma to obtain small EPG-representations for a number of graph classes (we will not give formal definitions of these graph classes; see [6]).

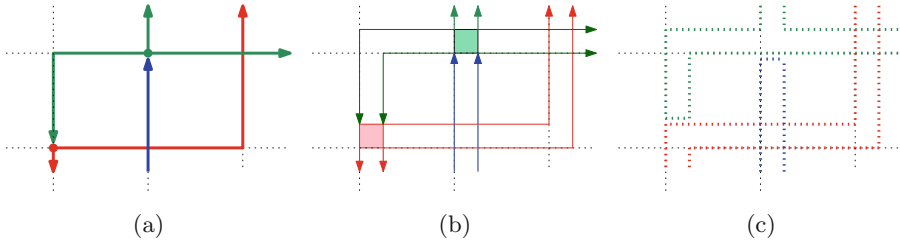


Fig. 5. Transforming an orthogonal drawing into an EPG-representation. For ease of reading we show the duplicated grid-line close to the original one.

Corollary 3. *All graphs of bounded treewidth (in particular, trees, outer-planar graphs and series-parallel graphs) have an EPG-representation in $O(n)$ area. Graphs of bounded genus have an EPG-representation in $O(n \log^2 n)$ area.*

Proof. Let G be one such graph for which we wish to obtain the EPG-representation. G may not be a 4-graph, but we can turn it into a 4-graph by *vertex-splitting*, defined as follows. Let v be a vertex with 5 or more neighbours w_1, \dots, w_d . Create a new vertex v' , which is adjacent to w_1, w_2, w_3 and v , and delete the edge (v, w_i) for $i = 1, 2, 3$. Observe that $\deg(v') = 4$ and $\deg(v)$ is reduced by 2, so sufficient repetition ensures that all vertex degrees are at most 4. Let H be the resulting graph, and observe that G is a minor of H .

Every vertex v of G gives rise to at most $\deg(v)/2$ new vertices in H , so H has at most $n + m$ vertices. Since graphs of bounded treewidth have $O(n)$ edges and graphs of bounded genus have $O(n)$ edges, therefore H has $O(n)$ vertices. Markov and Shi [16] argued that the splitting can be done in such a way that $tw(H) \leq tw(G) + 1$. It is also not hard to see that with a suitable way of splitting, one can ensure that in the case of bounded genus graphs the graph H obtained by splitting has the same genus.

By Leiserson’s construction [14], 4-graphs of bounded treewidth have an orthogonal drawing in $O(n)$ area and those of bounded genus have an orthogonal drawing in $O(n \log^2 n)$ area. □

For classes of 4-graphs, Lemma 4 and Leiserson’s construction [14] give directly the following stronger results:

Corollary 4. *Hereditary classes of 4-graphs that have balanced separators of size $O(n^\epsilon)$ with $\epsilon < 1/2$ have EPG-representation in $O(n)$ area. Hereditary classes of 4-graphs that have balanced separators of size $O(n^\epsilon)$ with $\epsilon > 1/2$ have EPG-representation in $O(n^{2\epsilon})$ area.*

The first bound in Corollary 4 is tight thanks to Theorem 2. The second bound is tight thanks to Corollary 2 and the fact that there are such classes of graphs which contain triangle-free graphs of pathwidth $\Omega(n^\epsilon)$, for example the class of finite 4-graphs that are subgraphs of the 3D integer grid with $\epsilon = 2/3$.

References

1. Alam, M.J., Bläsius, T., Rutter, I., Ueckerdt, T., Wolff, A.: Pixel and Voxel Representations of Graphs. In: Di Giacomo, E., Lubiw, A. (eds.) GD 2015. LNCS, vol. 9411, pp. 472–486. Springer, Cham (2015). <https://doi.org/10.1007/978-3-319-27261-0-39>
2. Alon, N., Seymour, P., Thomas, R.: A separator theorem for graphs with an excluded minor and its applications. In: Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing, STOC 1990, pp. 293–299. ACM, New York (1990)
3. Asinowski, A., Suk, A.: Edge intersection graphs of systems of paths on a grid with a bounded number of bends. *Discrete Appl. Math.* **157**(14), 3174–3180 (2009)
4. Biedl, T., Stern, M.: Edge-intersection graphs of k -bend paths in grids. *Discrete Math. Theor. Comput. Sci.* **12**(1), 1–12 (2010). (electronic journal)
5. Bodlaender, H.: A partial k -arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.* **209**(1–2), 1–45 (1998)
6. Diestel, R.: *Graph Theory*. GTM, vol. 173. Springer, Heidelberg (2017). <https://doi.org/10.1007/978-3-662-53622-3>
7. Dvorak, Z., Norin, S.: Treewidth of graphs with balanced separations. CoRR abs/1408.3869 (2014)
8. Epstein, D., Golubic, M.C., Morgenstern, G.: Approximation Algorithms for B_1 -EPG Graphs. In: Dehne, F., Solis-Oba, R., Sack, J.-R. (eds.) WADS 2013. LNCS, vol. 8037, pp. 328–340. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40104-6_29
9. Francis, M., Lahiri, A.: VPG and EPG bend-numbers of Halin graphs. *Discrete Appl. Math.* **215**, 95–105 (2016)
10. Golubic, M.C.: *Algorithmic Graph Theory and Perfect Graphs*, 2nd edn. Academic Press, New York (2004)
11. Golubic, M., Lipshteyn, M., Stern, M.: Edge intersection graphs of single bend paths on a grid. *Networks* **54**(3), 130–138 (2009)
12. Grohe, M., Marx, D.: On tree width, bramble size, and expansion. *J. Comb. Theory, Ser. B* **99**(1), 218–228 (2009)
13. Heldt, D., Knauer, K., Ueckerdt, T.: Edge-intersection graphs of grid paths: the bend-number. *Discrete Appl. Math.* **167**, 144–162 (2014)
14. Leiserson, C.: Area-efficient graph layouts (for VLSI). In: IEEE Symposium on Foundations of Computer Science (FOCS 1980), pp. 270–281 (1980)
15. Marcus, A., Spielman, D., Srivastava, N.: Interlacing families I: bipartite Ramanujan graphs of all degrees. In: Symposium on Foundations of Computer Science, FOCS. pp. 529–537. IEEE Computer Society (2013)
16. Markov, I., Shi, Y.: Constant-degree graph expansions that preserve treewidth. *Algorithmica* **59**(4), 461–470 (2011)
17. Mehrabi, S.: Approximation algorithms for independence and domination on B_1 -VPG and B_1 -EPG-graphs. CoRR abs/1702.05633 (2017)

Mixed Linear Layouts of Planar Graphs

Sergey Pupyrev^(✉)

University of Arizona, Tucson, AZ, USA
spupyrev@gmail.com

Abstract. A k -stack (respectively, k -queue) layout of a graph consists of a total order of the vertices, and a partition of the edges into k sets of non-crossing (non-nested) edges with respect to the vertex ordering. In 1992, Heath and Rosenberg conjectured that every planar graph admits a mixed 1-stack 1-queue layout in which every edge is assigned to a stack or to a queue that use a common vertex ordering.

We disprove this conjecture by providing a planar graph that does not have such a mixed layout. In addition, we study mixed layouts of graph subdivisions, and show that every planar graph has a mixed subdivision with one division vertex per edge.

1 Introduction

A *stack layout* of a graph consists of a linear order on the vertices and an assignment of the edges to *stacks*, such that no two edges in a single stack cross. A “dual” concept is a *queue layout*, which is defined similarly, except that no two edges in a single queue may be nested. The minimum number of stacks (queues) needed in a stack layout (queue layout) of a graph is called its *stack number* (*queue number*). Stack and queue layouts were respectively introduced by Ollmann [13] and Heath et al. [10, 11]. These are ubiquitous structures with a variety of applications, including complexity theory, VLSI design, bioinformatics, parallel process scheduling, matrix computations, permutation sorting, and graph drawing; see [7] for more details.

Stack and queue layouts have been extensively studied for planar graphs. The stack number of a graph, also known as *book thickness*, is one if and only if the graph is outerplanar [4]. The stack number of a graph G is at most two if and only if G is subhamiltonian, that is, a subgraph of a planar graph that has a Hamiltonian cycle [4]. More generally, all planar graphs have stack number at most four [18]. Similarly, every graph admitting a 1-queue layout is planar with an “arched leveled-planar” embedding [10]. Many subclasses of planar graphs have bounded queue number: Every tree has queue number one [11], outerplanar graphs have queue number at most two [10]; series-parallel graphs have queue number at most three [15], and planar 3-trees have queue number at most seven [17]. It is, however, an open question whether every planar graph have a constant queue number; Dujmović shows that planar graphs have queue number $\mathcal{O}(\log n)$ [6], improving an earlier result of $\mathcal{O}(\log^4 n)$ by Di Battista et al. [5].

Stack and queue layouts are generalized through the notion of a *mixed* layout, in which every edge is assigned to a stack or to a queue that is defined with respect to a common vertex ordering [11]. Such a layout is called an *s-stack q-queue* layout, if it utilizes s stacks and q queues. One reason for studying mixed stack and queue layouts is that they model the dequeue data structure, as a dequeue may be simulated by two stacks and one queue [1, 8]. Here we study mixed layouts of planar graphs.

In their seminal paper [11], Heath and Rosenberg make the following conjecture, which has hitherto been unresolved.

Conjecture 1 (Heath and Rosenberg [11]). *Every planar graph admits a mixed 1-stack 1-queue layout.*

In this paper we disprove the conjecture by providing a planar graph that does not have a 1-stack 1-queue layout.

Theorem 1. *There exists a planar graph that does not admit a mixed 1-stack 1-queue layout.*

We found, however, that mixed layouts are rather “powerful”. Our experimental evaluation indicates that *all* planar graphs with $|V| \leq 18$ vertices admit a 1-stack 1-queue layout. This is in contrast with pure stack and queue layouts: There exists a 11-vertex planar graph that requires three stacks, and there exists 14-vertex planar graphs that requires three queues. Thus a reasonable question is what subclasses of planar graphs admit a 1-stack 1-queue layout. Dujmović and Wood [8] consider graph subdivisions, that is, graphs created by replacing every edge of a graph by a path; they show that every planar graph has a 1-stack 1-queue subdivision with four division vertices per edge. We strengthen this result by showing that one division vertex per edge is sufficient.

Theorem 2. *Every planar graph admits a mixed 1-stack 1-queue subdivision with one division vertex per edge.*

Proof Ideas and Organization. Our construction of the counterexample for Conjecture 1 (presented in Sect. 2) is based on a sequence of gadgets — planar graphs that do not admit a mixed layout under certain conditions. We start with a relatively simple gadget whose linear layouts can be analyzed exhaustively; this gadget does not admit a mixed layout under fairly strong conditions. Several small gadgets are combined into a bigger one, that does not have a mixed layout under weaker conditions; these bigger gadgets are combined together to produce the final counterexample. We believe that such an approach is general and can be used for creating other lower bounds in the context of linear layouts.

Our technique for proving Theorem 2 (considered in Sect. 3) is quite different from the one used by Dujmović and Wood [8] for proving the earlier (weaker) result. We make use of the so-called concentric representation of planar graphs. While the existence of such a representation for a planar graph is known, we extend the representation by finding a suitable order for the vertices, and show

that all planar graphs admit the extended representation. We are not aware of any work that uses concentric representations in the context of linear layouts.

Section 4 concludes the paper with a discussion of our experiments, possible future directions, and interesting open problems.

Related Work. Although there exists numerous works on stack and queue layouts of graphs (refer to [7] for a detailed list of references), the concept of mixed layouts received much less attention. Heath and Rosenberg [11] suggest to study such generalized layouts and present Conjecture 1, which is a topic of this paper. Dujmović and Wood [8] investigate mixed layouts of graph subdivisions. They show that every graph G (not necessarily planar) has an s -stack q -queue subdivision with $\mathcal{O}(\log sn(G))$ or $\mathcal{O}(\log qn(G))$ vertices per edge, where $sn(G)$ and $qn(G)$ are the stack and queue numbers of G , respectively. Enomoto and Miyachi [9] improve the constants of the bounds for the numbers of division vertices per edge.

For the case of planar graphs, Dujmović and Wood [8] show that four division vertices per edge are sufficient to construct a mixed 1-stack 1-queue; the bound is improved by Theorem 2. Our result mimics the fact that every planar graph with one division vertex per edge has a 2-stack layout, as such graphs are bipartite [14]. Also related is a work by Auer [1] who study *dequeue* layouts of planar graphs, and prove that a planar graph admits a dequeue layout if and only if it contains a Hamiltonian path. Since a dequeue may be simulated by two stacks and one queue, such graphs also admit a 2-stack 1-queue layout. To the best of our knowledge, it is open whether every planar graph has a 2-stack 1-queue layout.

2 A Counterexample for Conjecture 1

A *vertex ordering* of a graph $G = (V, E)$ is a total order of the vertex set V . In a vertex ordering $<$ of G , let $L(e)$ and $R(e)$ denote the endpoints of an edge $e \in E$ such that $L(e) < R(e)$. Consider two edges $e, f \in E$. If $L(e) < L(f) < R(e) < R(f)$ then e and f *cross*, and if $L(e) < L(f) < R(f) < R(e)$ then e and f *nest*. In the latter case, we also say that e *covers* f . It is convenient to express the total order $<$ by permutation of vertices $[v_1, v_2, \dots, v_{|V|}]$, where $v_1 < v_2 < \dots < v_{|V|}$. This notion extends to a subset of vertices in the natural way. Thus, two edges, e and f , cross if the order is $[L(e), L(f), R(e), R(f)]$, and they nest if the order is $[L(e), L(f), R(f), R(e)]$. A *stack* (resp. *queue*) is a set of edges $E' \subset E$ such that no two edges in E' cross (nest). A *mixed* layout of a graph is a pair $(<, \{\mathcal{S}, \mathcal{Q}\})$, where $<$ is a vertex ordering of G , and $\{\mathcal{S}, \mathcal{Q}\}$ is a partition of E into a stack \mathcal{S} and a queue \mathcal{Q} .

Our counterexample for Conjecture 1 is depicted in Fig. 1. The graph, \overline{G} , is built from 19 copies of a gadget, H , by identifying two vertices, A and B . The graph consists of 173 vertices and 361 edges. Let us introduce some definitions for the graph. Every copy of gadget H consists of two *twins*, s and t , connected by a *twin edge*, (s, t) . Each pair of twins is connected by A, B , and seven degree-2 vertices, x_1, \dots, x_7 , that we call *connectors*. The set of connectors corresponding to s and t is denoted by $C_{s,t}$. Now we prove the main result of the section.

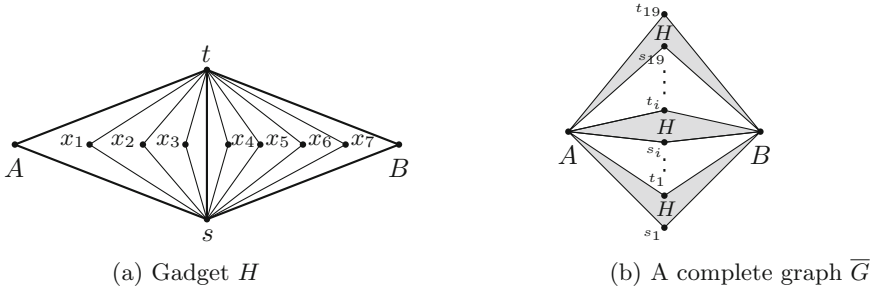


Fig. 1. A graph that does not admit a mixed 1-stack 1-queue layout.

Theorem 1. *There exists a planar graph that does not admit a mixed 1-stack 1-queue layout.*

Proof. The proof is by contradiction; we assume that there exists a mixed layout of graph \overline{G} shown in Fig. 1b. Using symmetry, we may assume that in the mixed layout of the graph $A < B$, $s_i < t_i$ for all $1 \leq i \leq 19$, and $s_1 < s_2 < \dots < s_{19}$. Let us analyze possible relative orderings of vertices A, B and two twins, s and t in a gadget H . It is easy to see that there are only six permutations of the vertices: (i) $[s, t, A, B]$; (ii) $[s, A, t, B]$; (iii) $[s, A, B, t]$; (iv) $[A, s, t, B]$; (v) $[A, s, B, t]$; (vi) $[A, B, s, t]$. Since graph \overline{G} contains 19 pairs of twins, there exist at least four twin pairs that form the same permutation with A and B . Therefore, to prove the claim of the theorem, it is sufficient to show impossibility of a mixed layout with four twin pairs forming the same permutation. Permutations (i) and (vi) are considered in Case 4, as they are symmetric. Permutations (ii) and (v) are considered in Case 2. Permutation (iii) is considered in Case 1. Permutation (iv) is considered in Case 3. \square

Before moving to the case analysis, we prove three lemmas that are common for the proofs of all the cases.

Lemma 1. *Assume that a vertex ordering of graph \overline{G} contains $[v_1, s, t, v_2]$ with edge $(v_1, v_2) \in \mathcal{Q}$ and twins s, t . Then the following holds:*

- 1a** *the order is $[v_1, s, t, v_2, x_1, x_2, x_3]$ or $[x_1, x_2, x_3, v_1, s, t, v_2]$ for some connectors $x_1, x_2, x_3 \in C_{s,t}$; that is, at least three of the connectors are either before v_1 or after v_2 in the order;*
- 1b** *$(s, x_i) \in \mathcal{S}$ and $(t, x_i) \in \mathcal{Q}$, or $(s, x_i) \in \mathcal{Q}$ and $(t, x_i) \in \mathcal{S}$ for some $x_i \in C_{s,t}$, $1 \leq i \leq 3$; that is, at least one of the connectors is adjacent to a queue edge and a stack edge.*

Proof. For the first part of the lemma, assume that three of the connectors corresponding to s and t are between v_1 and v_2 ; that is, $v_1 < x_5 < x_6 < x_7 < v_2$ for some connectors $x_5, x_6, x_7 \in C_{s,t}$. Since the graph induced by vertices s, t, x_5, x_6, x_7 is not 1-stack (that is, outerplanar), at least one of edges, (s, x_5) ,

$(s, x_6), (s, x_7), (t, x_5), (t, x_6), (t, x_7)$, is a queue edge. However, this edge is covered by $(v_1, v_2) \in \mathcal{Q}$, a contradiction.

For the second part of the lemma, assume the order is $[v_1, s, t, v_2, x_1, x_2, x_3]$ (the proof for the other order is symmetric). Suppose that none of the connectors is adjacent to both queue and stack edges. Hence, there are two connectors, say x_1 and x_2 , with edges assigned to a queue or to a stack. However, one of (s, x_1) and (t, x_2) is a queue edge, as the two edges cross. Similarly, one of edges (s, x_2) and (t, x_1) is a stack edge, as the edges are nested, a contradiction. \square

Lemma 2. *Assume that a vertex ordering of graph \overline{G} contains $[v_1, u_1, s, t, u_2, v_2]$ with edges $(v_1, v_2) \in \mathcal{Q}$, $(u_1, u_2) \in \mathcal{S}$ and twins s, t . Then \overline{G} does not admit a mixed layout.*

Proof. By Lemma 1a applied for vertices v_1, s, t, v_2 , there exists a connector $x \in C_{s,t}$ such that $x < v_1$ or $x > v_2$. By Lemma 1b, one of edges $(s, x), (t, x)$ is a stack edge. However, this edge crosses stack edge (u_1, u_2) , a contradiction; see Fig. 2a. \square

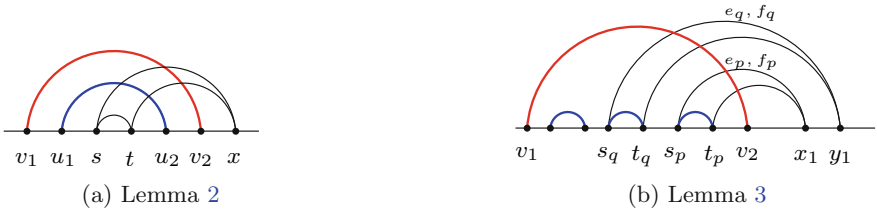


Fig. 2. Impossible configurations for a mixed layout of graph \overline{G} , as shown by Lemmas 2 and 3. Stack edges are blue and queue edges are red. (Color figure online)

Lemma 3. *Assume that for three pairs of twins $s_i, t_i, 1 \leq i \leq 3$, a vertex ordering of graph \overline{G} contains $[v_1, s_i, t_i, v_2]$, where edge $(v_1, v_2) \in \mathcal{Q}$. Then \overline{G} does not admit a mixed layout.*

Proof. Notice that all twin edges, (s_i, t_i) for $1 \leq i \leq 3$, are stack edges, as they are covered by $(v_1, v_2) \in \mathcal{Q}$. Moreover, the edges do not nest each other, as otherwise the two nested edges together with (v_1, v_2) form a configuration as in Lemma 2, which is impossible. Thus, we have three non-nested pairs of twins, that is, up to renumbering the order is $[v_1, s_1, t_1, s_2, t_2, s_3, t_3, v_2]$.

Let us apply Lemma 1a for the three pairs of twins and edge $(v_1, v_2) \in \mathcal{Q}$. There are two triples of connectors, $x_1, x_2, x_3 \in C_{s_p, t_p}$ and $y_1, y_2, y_3 \in C_{s_q, t_q}$ for $p, q \in \{1, 2, 3\}$, that are all either before v_1 or after v_2 in the order. Without loss of generality, we assume $v_2 < x_j$ and $v_2 < y_j$ for all $1 \leq j \leq 3$.

By Lemma 1b applied for twins s_p and t_p , one of the connectors, say x_1 , is adjacent to a queue edge, e_p , and to a stack edge, f_p . Similarly, a connector of

s_q and t_q , say y_1 , is adjacent to a queue edge, e_q , and to a stack edge, f_q ; see Fig. 2b. However, it is not possible to assign these four edges to \mathcal{S} and \mathcal{Q} : If $x_1 < y_1$, then the two queue edges, e_p and e_q , nest; If $x_1 > y_1$, then the two stack edges, f_p and f_q , cross. \square

Now we are ready to analyze the cases proving Theorem 1.

Case 1 ($sABt$). Assume that the vertex ordering is $[s_i, A, B, t_i]$ for twins s_i, t_i for all $1 \leq i \leq 4$. Then graph \overline{G} does not admit a mixed layout.

Proof. Let us assume that the vertex ordering is $[s_1, s_2, s_3, s_4, A, B, d_1, d_2, d_3, d_4]$, where $d_i \in \{t_1, t_2, t_3, t_4\}$ for all $1 \leq i \leq 4$. Note that \overline{G} contains edges (A, d_i) and (B, d_i) for all $1 \leq i \leq 4$.

Start with a pair of edges (A, d_4) and (B, s_1) ; since they cross, one of the edges is a queue edge. Without loss of generality, we may assume that $(A, d_4) \in \mathcal{Q}$. Hence, all edges covered by (A, d_4) are stack edges; that is, $(B, d_3), (B, d_2), (B, d_1) \in \mathcal{S}$. It follows that all edges crossing the three edges are in the queue: $(A, d_2), (A, d_1) \in \mathcal{Q}$; see Fig. 3a.

Now let s_x be a twin of d_2 . Edge (s_x, d_2) is a stack edge since it covers $(A, d_1) \in \mathcal{Q}$. However, (s_x, d_2) crosses $(B, d_3) \in \mathcal{S}$, a contradiction. \square



Fig. 3. An illustration for the proofs of cases of Theorem 1. Stack edges are blue and queue edges are red. (Color figure online)

Case 2 ($AsBt$). Assume that the vertex ordering is $[A, s_i, B, t_i]$ for twins s_i, t_i for all $1 \leq i \leq 4$. Then graph \overline{G} does not admit a mixed layout.

Proof. Let us assume that the vertex ordering is $[A, s_1, s_2, s_3, s_4, B, d_1, d_2, d_3, d_4]$, where $d_i \in \{t_1, t_2, t_3, t_4\}$ for all $1 \leq i \leq 4$.

Suppose that $e = (A, d_2)$ is a stack edge. Then $(B, d_3) \in \mathcal{Q}$, as it crosses e . This is impossible, as twin edge (s_x, d_4) (for some $1 \leq x \leq 4$) crosses a stack edge, (A, d_2) , and covers a queue edge, (B, d_3) . Hence, (A, d_2) is a queue edge.

Since $(A, d_2) \in \mathcal{Q}$, all nested edges are in the stack; in particular, $(s_1, B) \in \mathcal{S}$ and $(s_y, d_1) \in \mathcal{S}$, where s_y is the twin of d_1 . Notice that in order for edges (s_1, B) and (s_y, d_1) to be non-crossing, s_y should be equal to s_1 ; see Fig. 3b. It follows that edge (B, d_2) is a queue edge since it crosses (s_1, d_1) . Finally, we observe that the twin edge of d_3 , (s_z, d_3) for some $2 \leq z \leq 4$, crosses a stack edge, (s_1, d_1) , and covers a queue edge, (B, d_2) , which is not possible. \square

Case 3 (*AstB*). Assume that the vertex ordering is $[A, s_i, t_i, B]$ for twins s_i, t_i for all $1 \leq i \leq 4$. Then graph \overline{G} does not admit a mixed layout.

Proof. Let us assume that the vertex ordering is $[A, d_1, d_2, \dots, d_8, B]$, where $s_i, t_i \in \{d_1, \dots, d_8\}$ for all $1 \leq i \leq 4$.

Since edges (A, d_8) and (d_1, B) cross, one of them is a queue edge. Without loss of generality, we may assume $(A, d_8) \in \mathcal{Q}$. Consider seven vertices d_1, d_2, \dots, d_7 . It is easy to see that they form three pairs of twins (while the fourth pair is formed with d_8). By Lemma 3 applied for $(A, d_8) \in \mathcal{Q}$ and the twins, it is impossible. \square

Case 4 (*ABst*). Assume that the vertex ordering is $[A, B, s_i, t_i]$ for twins s_i, t_i for all $1 \leq i \leq 4$. Then graph \overline{G} does not admit a mixed layout.

Proof. Let us assume that the vertex ordering is $[A, B, d_1, d_2, \dots, d_8]$, where $s_i, t_i \in \{d_1, \dots, d_8\}$ for all $1 \leq i \leq 4$.

Suppose that edge $e = (A, d_7) \in \mathcal{Q}$; then edge $(B, d_6) \in \mathcal{S}$, as it is covered by e . Additionally, we have five vertices, d_1, d_2, d_3, d_4, d_5 , which form at least one pair of twins. This pair of twins together with $(A, d_7) \in \mathcal{Q}$ and $(B, d_6) \in \mathcal{S}$ form a configuration as in Lemma 2, which is impossible. Therefore, edge $(A, d_7) \in \mathcal{S}$ and the crossing edge, (B, d_8) , is a queue edge.

Consider vertices d_1, \dots, d_7 . There are three pairs of twins formed by the vertices; all of the pairs are covered by $(B, d_8) \in \mathcal{Q}$, contradicting Lemma 3. \square

3 Mixed Layouts of Planar Subdivisions

In this section we prove Theorem 2. To this end, we utilize a special representation of a planar graph, which is called *ordered concentric*. In such a representation, the vertices of a graph are laid out on a set of circles around a specified origin vertex, so that each circle contains exactly the vertices with the same graph-theoretic distance to the origin; see Fig. 4. To construct such a representation, we begin with an arbitrary vertex of the graph as the origin, and consider a planar embedding of the graph with the origin on the outer face. The layers of vertices are formed by a breadth-first search starting at the origin, and the edges are routed without crossings and connect vertices of the same layer or vertices of two consecutive layers. Formally an ordered concentric representation is defined next. We assume that for a graph $G = (V, E)$, $dist_G(u, v)$ is the graph-theoretic distance between vertices $u, v \in V$.

Definition 1. Let $G = (V, E)$ be a connected planar graph with a specified vertex $v^* \in V$. An **ordered concentric representation of G with the origin v^*** , denoted by Γ^o , is a drawing of G with the following properties:

- (i) The drawing is planar with vertex v^* lying on the outer face;
- (ii) $V_i = \{x \in V \mid dist_G(v^*, x) = i\}$ for all $0 \leq i \leq k$ and some $k \in \mathbb{N}$. The set V_i is called the **i -th level** of G . For every level V_i , $1 \leq i \leq k$, the vertices of V_i are arranged in a sequence x_1, x_2, \dots, x_r with $|V_i| = r$.

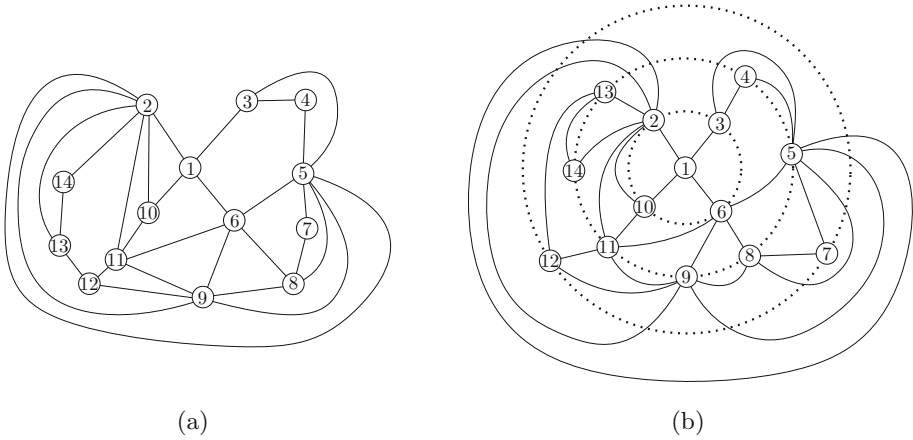


Fig. 4. (a) A plane graph and (b) its ordered concentric representation with the origin $v^* = 1$. Concentric circles are shown dashed.

In the drawing, the vertices are laid out on a closed curve in the order; the curve is called the i -th circle. The vertices of V_j , $j < i$ are located inside the area bounded by the i -th circle, while the vertices of V_j , $j > i$ are located outside the area bounded by the i -th circle.

- (iii) For every edge $(u, w) \in E$ with $u \in V_i$ and $w \in V_j$, it holds that either $i = j$, in which case the edge is called a **level edge**, or $|i - j| = 1$, corresponding to a **non-level edge**.

Every level edge $(u, w) \in E$, $u, w \in V_i$ is realized as a curve routed outside the i -th circle. Every non-level edge $(u, w) \in E$, $u \in V_i$, $w \in V_{i+1}$ is realized as a curve consisting of at most two pieces: the first (required) piece is routed between the i -th and the $(i + 1)$ -th circles, and the second (optional) piece is routed outside the $(i + 1)$ -th circle.

Notice that the notion of concentric representations is related to radial drawings [2]. The main difference is that in radial drawings “monotonicity” of edges is required; equivalently, every edge shares at most one point with a circle in a radial drawing. In contrast, some edges of a concentric representation may cross a circle multiple times; for example, see an edge $(5, 8)$ in Fig. 4b. As a result, a radial drawing may not exist for a planar graph, while an ordered concentric representation can always be constructed as shown by Lemma 4.

Another closely related concept is a (non-ordered) concentric representation of a planar graph [12, 16]. Such a representation is defined similarly, except that the vertices of each level form a cyclic sequence and the origin vertex is not required to lie on the outer face. In a sense, Definition 1 provides a refinement of a concentric representation, as it dictates the (non-cyclic) order of the vertices of every level. The next lemma shows that every planar graph admits an ordered concentric representation.

Lemma 4. *For every connected planar graph $G = (V, E)$ and every $v^* \in V$, there exists an ordered concentric representation of G with the origin v^* .*

Proof. We start by constructing a breadth-first search tree, T , of G rooted at v^* . Consider an arbitrary combinatorial embedding of G (that is, cyclic orders of edges around each vertex), and draw T on a set of horizontal lines respecting the planar embedding. A line with y -coordinate $= i$ contains vertices V_i , $0 \leq i \leq k$ with $k = \max_{u \in V} \text{dist}_G(v^*, u)$. The order of the vertices along each line is defined by the embedding of G . Notice that the drawing is planar and satisfies Definition 1. Here the circles are formed by connecting the leftmost and the rightmost vertices of each horizontal line with a curve surrounding the drawing; see Fig. 5a. All the edges of T are drawn as straight-line segments between consecutive levels, that is, they are non-level edges. Next we show how to draw the remaining edges of E while preserving the properties of Definition 1.

To this end, we maintain the following invariant: For every face f of the currently drawn graph, H , there exists a vertical line segment of length $\varepsilon > 0$ such that every vertex $u \in f$ can be connected to every point of the segment via a curve, which is monotone in the y -direction, while avoiding crossing with the edges of H . Since the drawing of T defines only one face, it is clear that the segment with endpoints (x, k) and $(x, k + 1)$ (for an arbitrary $x \in \mathbb{R}$) satisfies the invariant; see Fig. 5a. Let us show how to draw the next edge. Assume that an edge, $(u, w) \in E$, belongs to a face f of H . Due to the invariant, there exists a line segment with endpoints $p_0 = (x, y_0)$ and $p_1 = (x, y_1)$ (for some $x, y_0, y_1 \in \mathbb{R}$) that is reachable from both u and w . We identify point $p = (x, (y_0 + y_1)/2)$ on the segment and route the edge along the curves connecting u to p and then p to w . The edge splits f into two faces such that the condition of the invariant can be satisfied using segments p_0, p and p, p_1 ; see Fig. 5b. To complete the proof,

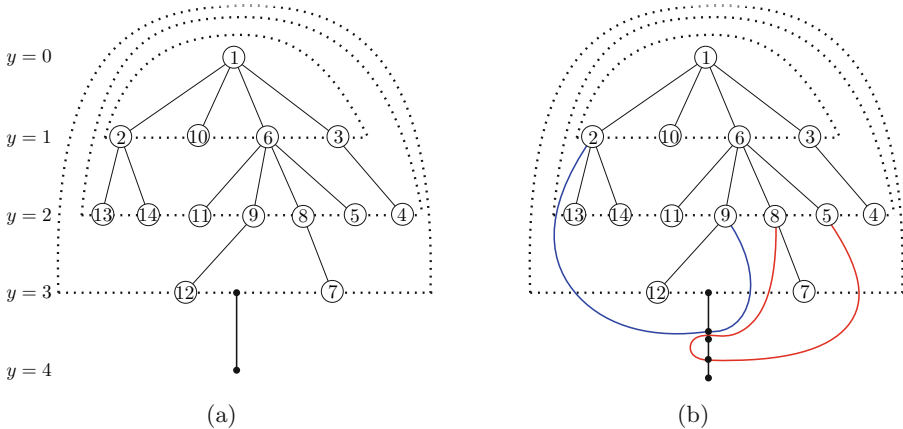


Fig. 5. (a) A starting point for construction an ordered concentric representation for graph shown in Fig. 4a, as described in Lemma 4. (b) Maintaining the invariant of Lemma 4 while drawing edge $(2, 9)$ and then edge $(5, 8)$.

we observe that the edge also satisfies Definition 1. If $dist_G(v^*, u) = dist_G(v^*, w)$, then (u, w) is a level edge. Otherwise, if $|dist_G(v^*, u) - dist_G(v^*, w)| = 1$, then (u, w) is a non-level edge represented by a two-piece curve. \square

Now we are ready to prove the main result of the section. Our construction of a mixed layout for a given graph G is as follows. We start with an ordered concentric representation, Γ^o , of G , which is created by a breadth-first search starting at an arbitrary vertex v^* . We distinguish three types of edges in Γ^o : (a) level edges with both endpoints belonging to the same level of Γ^o , (b) *short* non-level edges whose curves are routed between consecutive levels of Γ^o , and (c) *long* non-level edges whose curves cross some circles of Γ^o . Our goal is to keep the edges of type (a) in the stack and the edges of type (b) in the queue. The edges of type (c) shall be subdivided into a stack edge and a queue edge. The order of vertices in the mixed layout is constructed from the levels of Γ^o : we place the origin v^* , which is followed by the vertices of V_1 (in the order given by the ordered concentric representation), followed by the vertices of V_2 etc. The correctness of the mixed layout follows from the planarity of Γ^o ; see Fig. 6. Next we provide a formal proof.

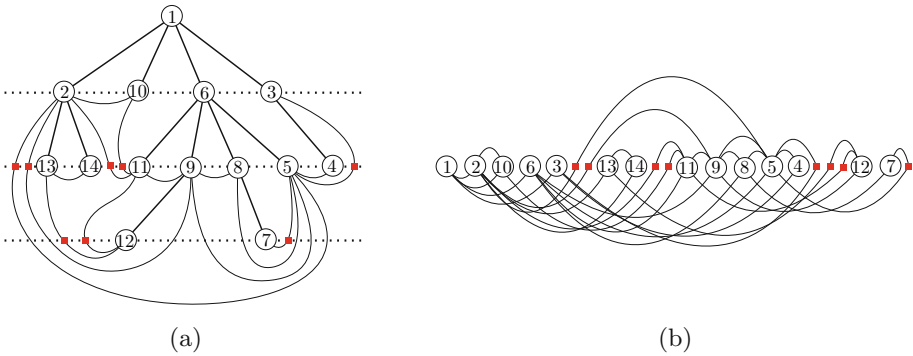


Fig. 6. (a) An ordered concentric representation for graph shown in Fig. 4a. Red squares are subdivision vertices introduced for long non-level edges. (b) A mixed 1-stack 1-queue layout of the graph constructed as described in Theorem 2. (Color figure online)

Theorem 2. *Every planar graph admits a mixed 1-stack 1-queue subdivision with one division vertex per edge.*

Proof. Let $G = (V, E)$ be a planar graph and $v^* \in V$. We assume that G is connected; otherwise, each connected component of G can be processed individually. Using Lemma 4, construct an ordered concentric representation Γ^o of the graph with the origin v^* and a set of levels V_i , $0 \leq i \leq k$ for some $k \geq 0$.

Consider an edge $(u, w) \in E$ with $u \in V_i$ and $w \in V_j$. Since the levels are constructed with a breadth-first search, it holds that $i = j$ or $|i - j| = 1$. Let $P_{u,w}$

be the (ordered) sequence of levels of Γ^o such that the corresponding circles share a point with the curve realizing edge (u, w) . By Definition 1, $E = E_a \cup E_b \cup E_c$, where

- $E_a = \{(u, w) : u, w \in V_i \text{ for some } 1 \leq i \leq k \text{ with } P_{u,w} = (V_i, \dots, V_i) \text{ in } \Gamma^o\}$;
- $E_b = \{(u, w) : u \in V_i, w \in V_{i+1} \text{ with } P_{u,w} = (V_i, V_{i+1}) \text{ in } \Gamma^o\}$;
- $E_c = \{(u, w) : u \in V_i, w \in V_{i+1} \text{ with } P_{u,w} = (V_i, V_{y_1}, \dots, V_{y_d}, V_{i+1}) \text{ in } \Gamma^o \text{ for some } d \geq 1, y_1 = i + 1 \text{ and } y_t > i + 1 \text{ for } 2 \leq t \leq d\}$.

We construct a subdivision $G^s = (V^s, E^s)$ as follows. For an edge $e = (u, w) \in E_c$, consider the crossing point, $y_1(e)$, between the $(i + 1)$ -th circle and the curve realizing e in Γ^o (which corresponds to the second element, V_{y_1} , of $P_{u,w}$). Let

$$V^s = V \cup \bigcup_{e \in E_c} \{y_1(e)\} \text{ and } E^s = E_a \cup E_b \cup E_{c_1} \cup E_{c_2}, \text{ where}$$

$$E_{c_1} = \bigcup_{e \in E_c} \{(u, y_1(e))\} \text{ and } E_{c_2} = \bigcup_{e \in E_c} \{(y_1(e), w)\}.$$

In order to construct a mixed layout of G^s , we use the following order:

$$\sigma = (v^*, x_1^1, x_2^1, \dots, x_{r_1}^1, x_1^2, x_2^2, \dots, x_{r_2}^2, x_1^3, x_2^3, \dots, x_{r_3}^3, \dots),$$

where $\{x_1^i, x_2^i, \dots, x_{r_i}^i\} = V_i^s \supseteq V_i$ are the vertices of the i -th level of G^s in the order given by Γ^o . All edges of E_a and E_{c_2} are stack edges; that is, $e \in \mathcal{S}$ for every $e \in E_a \cup E_{c_2}$. All the remaining edges are queue edges; that is, $e \in \mathcal{Q}$ for every $e \in E_b \cup E_{c_1}$. Next we prove the correctness of the construction.

Let us show that all stack edges are crossing-free with respect to the specified order, σ . Assume two edges, $(u_1, w_1), (u_2, w_2) \in \mathcal{S}$, cross each other, that is $u_1 < u_2 < w_1 < w_2$ with respect to σ . Observe that all edges in the stack are the edges of the same level in the ordered concentric representation. Thus, $u_1, w_1 \in V_i^s$ and $u_2, w_2 \in V_j^s$ for some $0 \leq i, j \leq k$. However, the levels are arranged consecutively in σ , which means that $i = j$ and two edges of the same level cross. This is impossible, as all vertices of the same level of Γ^o and the corresponding level edges form an outerplanar graph by Definition 1.

Finally, let us show that all queue edges are non-nested with respect to σ . Assume that two edges, $(u_1, w_1), (u_2, w_2) \in \mathcal{Q}$, nest each other so that $u_1 < u_2 < w_2 < w_1$. Since the queue edges belong to consecutive levels in the ordered concentric representation, it holds that $u_1 \in V_i^s, w_1 \in V_{i+1}^s, u_2 \in V_j^s, w_2 \in V_{j+1}^s$ for some $0 \leq i, j \leq k$. Since the levels do not overlap in σ , it holds that $i = j$. Hence, the two edges are routed between the same consecutive levels, V_i^s and V_{i+1}^s , and therefore, cross each other, which violates the planarity of Γ^o . \square

4 Discussion

In this paper we resolved a conjecture by Heath and Rosenberg [11] by providing a graph that does not admit a mixed 1-stack 1-queue layout. The graph contains

173 vertices, and a reasonable question is what is the size of the smallest counterexample. In an attempt to answer the question, we implemented an exhaustive search algorithm¹ (based on the SAT formulation of the linear embedding problem suggested by Bekos et al. [3]) and tested *all* 977,526,957 maximal planar graphs with $|V| \leq 18$. It turns out that all such graphs have a mixed 1-stack 1-queue layout. The evaluation suggests that mixed layouts are more “powerful” than pure stack and queue layouts, as there exist fairly small graphs that do not admit 2-stack and 2-queue layouts. The smallest planar graph requiring three stacks contains 11 vertices, and the smallest planar graph requiring three queues contains 14 vertices; see Figs. 7a and b. We were able to find a smaller counterexample for Conjecture 1; see Fig. 7c. This instance consists of $|V| = 37$ vertices and $|E| = 77$ edges, and has a similar structure as the graph in Theorem 1. However, showing that the graph does not admit a mixed layout requires significantly more effort.

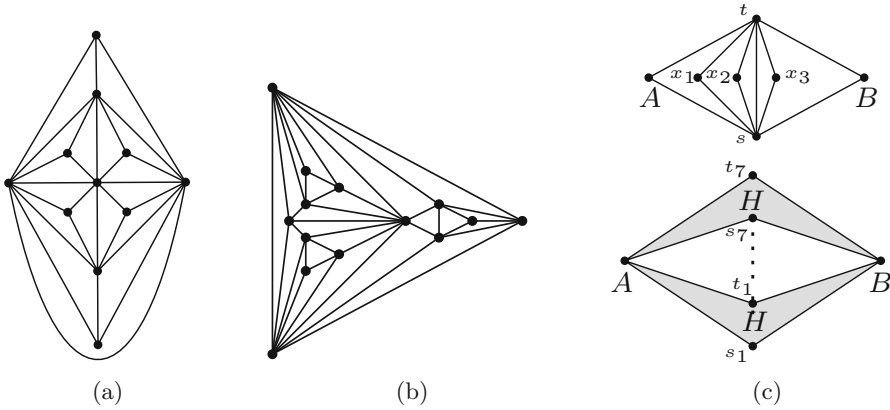


Fig. 7. The smallest planar graphs that require (a) 3 stacks, (b) 3 queues. (c) A graph with $|V| = 37$ and $|E| = 77$ that does not admit a mixed layout.

An interesting future direction is to consider *bipartite* planar graphs. We noticed that all our counterexamples contain triangles, which seem to be important for non-embeddability. Based on our experiments, we conjecture that every bipartite planar graph admits a mixed 1-stack 1-queue layout. Such a result would strengthen Theorem 2, as a subdivision of a graph with one vertex per edge is clearly bipartite. Also observe that a pure 2-stack layout exists for every bipartite planar graph, as shown by Overbay [14].

Conjecture 2. *Every bipartite planar graph admits a mixed 1-stack 1-queue layout.*

¹ An online tool and the source code for testing linear embeddability of graphs is available at <http://be.cs.arizona.edu>.

Another future direction is to study mixed s -stack q -queue layouts of planar graphs. What are possible values of s and q such that there exists a mixed s -stack q -queue layout for every planar graph? By the result of Yannakakis [18], we know that $s = 4, q = 0$ is a valid option, while Theorem 1 shows that $s = 1, q = 1$ is not sufficient. Here it is worth mentioning a result by Auer [1] who shows that every planar graph with a Hamiltonian path admits a mixed layout with $s = 2$ and $q = 1$. However it is open whether there exists some $s > 0$ and $q > 0$ with $2 < s + q \leq 4$ realizing all planar graphs.

References

1. Auer, C.: Planar graphs and their duals on cylinder surfaces. Ph.D. thesis, Universität Passau (2014)
2. Bachmaier, C., Brandenburg, F.J., Forster, M.: Radial level planarity testing and embedding in linear time. *J. Graph Algorithms Appl.* **9**(1), 53–97 (2005)
3. Bekos, M.A., Kaufmann, M., Zielke, C.: The book embedding problem from a SAT-solving perspective. In: Di Giacomo, E., Lubiw, A. (eds.) GD 2015. LNCS, vol. 9411, pp. 125–138. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27261-0_11
4. Bernhart, F., Kainen, P.C.: The book thickness of a graph. *J. Comb. Theor. Ser. B* **27**(3), 320–331 (1979)
5. Di Battista, G., Frati, F., Pach, J.: On the queue number of planar graphs. *SIAM J. Comput.* **42**(6), 2243–2285 (2013)
6. Dujmović, V.: Graph layouts via layered separators. *J. Comb. Theor. Ser. B* **110**, 79–89 (2015)
7. Dujmović, V., Wood, D.R.: On linear layouts of graphs. *Discrete Math. Theor. Comput. Sci.* **6**(2), 339–358 (2004)
8. Dujmović, V., Wood, D.R.: Stacks, queues and tracks: layouts of graph subdivisions. *Discrete Math. Theor. Comput. Sci.* **7**, 155–202 (2005)
9. Enomoto, H., Miyachi, M.: Stack-queue mixed layouts of graph subdivisions. In: *Forum on Information Technology*, pp. 47–56 (2014)
10. Heath, L.S., Leighton, F.T., Rosenberg, A.L.: Comparing queues and stacks as machines for laying out graphs. *SIAM J. Discrete Math.* **5**(3), 398–412 (1992)
11. Heath, L.S., Rosenberg, A.L.: Laying out graphs using queues. *SIAM J. Comput.* **21**(5), 927–958 (1992)
12. Hromkovič, J.: *Communication Complexity and Parallel Computing*. Springer, Heidelberg (2013). <https://doi.org/10.1007/978-3-662-03442-2>
13. Ollmann, L.T.: On the book thicknesses of various graphs. In: *Southeastern Conference on Combinatorics, Graph Theory and Computing*. vol. 8, p. 459 (1973)
14. Overbay, S.B.: *Generalized book embeddings*. Ph.D. thesis, Colorado State University (1998)
15. Rengarajan, S., Veni Madhavan, C.E.: Stack and queue number of 2-trees. In: Du, D.-Z., Li, M. (eds.) COCOON 1995. LNCS, vol. 959, pp. 203–212. Springer, Heidelberg (1995). <https://doi.org/10.1007/BFb0030834>
16. Ullman, J.D.: *Computational Aspects of VLSI*. Computer Science Press, Rockville (1984)
17. Wiechert, V.: On the queue-number of graphs with bounded tree-width. *Electr. J. Comb.* **24**(1), P1.65 (2017)
18. Yannakakis, M.: Embedding planar graphs in four pages. *J. Comput. Syst. Sci.* **38**(1), 36–67 (1989)

Upward Partitioned Book Embeddings

Hugo A. Akitaya¹(✉), Erik D. Demaine², Adam Hesterberg²,
and Quanquan C. Liu²(✉)

¹ Tufts University, Medford, MA, USA
hugo.alves_akitaya@tufts.edu

² Massachusetts Institute of Technology, Cambridge, MA, USA
{edemaine,achester,quanquan}@mit.edu

Abstract. We analyze a directed variation of the book embedding problem when the page partition is prespecified and the nodes on the spine must be in topological order (upward book embedding). Given a directed acyclic graph and a partition of its edges into k pages, can we linearly order the vertices such that the drawing is upward (a topological sort) and each page avoids crossings? We prove that the problem is NP-complete for $k \geq 3$, and for $k \geq 4$ even in the special case when each page is a matching. By contrast, the problem can be solved in linear time for $k = 2$ pages when pages are restricted to matchings. The problem comes from Jack Edmonds (1997), motivated as a generalization of the map folding problem from computational origami.

1 Introduction

Book Embeddings. Bernhart and Keinen [BK79] first introduced the concept of *book embeddings* and *book thickness* of graphs in 1979. Since then, book embeddings and book thickness have been widely studied as natural geometric invariants in directed and undirected graphs with applications in graph drawing and graph algorithms. Book embeddings (also studied under the name of *stack layouts* [CLR87, HP97, HPT99, HP99]) have applications in VLSI design, fault-tolerant processing, parallel process scheduling, sorting networks, and parallel matrix computations [CLR87, HLR92, HR92].

Given an undirected graph $G = (V, E)$, where $|V| = n$ and $|E| = m$, a *book embedding* consists of

1. a linear ordering π of the vertices V , defining an embedding of the vertices into the *spine* (a line in the plane); and
2. a disjoint partition of the edges E into sets, so that each set of the partition can be embedded into a *page* (half-plane bounded by the spine) without intersection between the edges on each page.

The pages join together at the spine to form a *book*. The *book thickness* of a graph G is the minimum number k of pages in any book embedding of G .

Much of the previous research on book embedding (on undirected graphs) focuses on the book thickness of particular graph classes such as complete

bipartite graphs [ENO97] and planar graphs (which have an upper bound of 4 pages) [Yan89, BBKR15, BGR16]. Graphs of book thickness 1 turn out to have a simple characterization as (exactly) outerplanar graphs [BK79], and such graphs can be recognized in linear time [Wie86]. By contrast, graphs with a 2-page book embedding are exactly the sub-Hamiltonian graphs [BK79], and recognizing such graphs is NP-complete [Wig82].

Directed Graphs. Motivated by many of the same applications, book embedding has been generalized to directed graphs. For directed acyclic graphs (DAGs), an *upward book embedding* is a book embedding such that the linear ordering of the vertices on the spine is in topological order [HP99, HPT99]. For general digraphs, *oriented book embeddings* [MK16] require that all arcs embedded into a page (or the spine) must agree in orientation (pointed up or down with respect to the order on the spine).

Research in upward book embedding includes combinatorial results for classes of DAGs such as trees, cycles, or paths by using characteristics of the underlying undirected graph [HP99]. Furthermore, more recent research studied the book embedding of directed planar graphs [FFR11]. As in the undirected case, there is a linear-time algorithm to determine whether a DAG has a 1-page upward book embedding [HP99] (although the algorithm is very different from the 1-page book embedding algorithm applied to the DAG's underlying undirected graph). Furthermore, determining whether a DAG has a 6-page upward book embedding is NP-complete [HP99]. There is a linear-time algorithm for 2-page upward book embedding of planar directed series-parallel graphs [DGDLW06]. Note that undirected series-parallel graphs are necessarily sub-Hamiltonian. For graphs with cycles, oriented book embeddings and oriented book thickness have been found for several graph classes including cycles and oriented trees [MK16].

See [DW04, DW05] for more detailed citations lists and surveys about book embeddings and linear graph layouts for both directed and undirected graphs.

Partitioned Problem. In the *partitioned* book embedding problem, we are *given* the partition of edges into pages. This variation eliminates one of the previous combinatorial aspects of finding a book embedding (namely finding the partition of edges), and leaves only finding the vertex order on the spine. Intuitively, this problem should be simpler. Indeed, there is a linear-time algorithm for determining whether a given edge partition can result in a 2-page book embedding of an undirected graph [HN14, ABD12]. Nonetheless, the partitioned k -page book embedding problem is NP-complete for undirected graphs where $k \geq 3$ [ALN15]. In their proof, the gadgets only work with undirected edges and therefore cannot directly be applied to upward book embedding.

In terms of partitioned book embedding problems where the edges in a partition form a matching, [Hos12] showed that for undirected graphs, it is NP-hard to find the ordering of vertices given unbounded number of pages (i.e. unbounded number of partitions). Although the reduction in [Hos12] is also from BETWEENNESS (defined in Definition 1), the techniques used are simpler and different from ours since they consider undirected graphs and allow unbounded number of partitions of edges.

Table 1. Summary of known and new results in partitioned book embedding. New results are written in bold.

Type	$k = 1$	$k = 2$	$k = 3$	$k \geq 4$
Undirected	$O(n)$ [BK79]	$O(n)$ [HN14]	NP-complete [ALN15]	NP-complete [ALN15]
Upward	$O(n)$ [HP99]	<i>OPEN</i>	NP-complete [Theorem 1]	NP-complete [Theorem 1]
Matching	$O(n)$ [HP99]	$O(n)$ [Theorem 3]	<i>OPEN</i>	NP-complete [Theorem 2]

Our Results. In this paper, we study the natural combination of the partitioned variation (where we are given the partition of edges into pages) with upward book embedding of DAGs, which has not been considered before (UPWARD PARTITIONED k -PAGE BOOK EMBEDDING). We prove that the resulting problem is NP-complete for any $k \geq 3$. Our hardness proof techniques also apply to a special case of this problem, called UPWARD MATCHING-PARTITIONED k -PAGE BOOK EMBEDDING, where only disjoint edges map to each page (forming a matching). For this special case, we show NP-hardness for $k \geq 4$ and that book embedding can be solved in linear time for $k = 1$ page or $k = 2$ pages. Table 1 puts these results in context with previous results.

UPWARD MATCHING-PARTITIONED 4-PAGE BOOK EMBEDDING is in fact motivated by the (nonsimple) map folding problem, posed by Jack Edmonds in 1997 (personal communication with E. Demaine); see [ABD+04,DLM12]. Edmonds showed that the problem of finding a flat folded state of an $m \times n$ grid crease pattern, with specified mountains and valleys, reduces to exactly this type of book embedding problem, with the $k = 4$ pages corresponding to the four compass directions of a square. Furthermore, $1 \times n$ and $2 \times n$ map folding reduce to the problems with $k = 2$ and $k = 3$ pages. Algorithms for solving UPWARD MATCHING-PARTITIONED k -PAGE BOOK EMBEDDING are thus of particular interest because they solve the long-standing map folding open problem as well.

In Sect. 2, we formally define our book embedding models. In Sect. 3, we prove NP-completeness for UPWARD PARTITIONED 3-PAGE BOOK EMBEDDING. Finally, in Sect. 4, we show that UPWARD MATCHING-PARTITIONED BOOK EMBEDDING can be solved in linear time for 2 pages and is NP-complete for 4 pages.

2 Definitions

We define the UPWARD PARTITIONED k -PAGE BOOK EMBEDDING (UPBE- k) problem similarly to the definition for PARTITIONED k -PAGE BOOK EMBEDDING as given in [ALN15]. Specifically, we are given a directed acyclic graph (DAG) $G = (V, E)$ and a partition of the edges in E : $P = \{E_1, E_2, \dots, E_k\}$ where

$E_1 \dot{\cup} E_2 \dot{\cup} \dots \dot{\cup} E_k = E$, where $\dot{\cup}$ denotes disjoint union. The goal is to determine whether G can be embedded in a k -page book such that the ordering π of the vertices on the spine is topologically sorted and each $E_i \in P$ lies in a separate page.

The UPWARD MATCHING-PARTITIONED k -PAGE BOOK EMBEDDING (UMPBE- k) problem is the special case of UPBE- k in which every edge partition $E_i \in P$ forms a directed matching, that is, has at most one edge incident to each vertex in G .

For a given upward partitioned book embedding instance $G = (V, E, P)$, let π represent a valid ordering of V on the spine where a valid ordering is one that satisfies the constraints on every page (e.g. non-crossing edges) and follows topological order. As stated previously, π is also a valid topological sorting of V . We write $\pi(x) < \pi(y)$ (resp., $\pi(x) > \pi(y)$) if node $x \in V$ comes earlier/before (resp., later/after) $y \in V$ in π . For ease of wording, we will assign colors to edge partitions and refer to edges within each partition to have a particular color.

3 UPBE is NP-Complete

We show that UPBE- k is NP-hard via a reduction from the NP-complete problem BETWEENNESS. The problem BETWEENNESS is defined as follows.

Definition 1 (Betweenness [Opa79]). *We are given a set L of n elements and a set C of m ordered triples where each member of a triple is a member of L . Let ϕ be a total ordering of the elements in L . An ordered triple, $\langle a, b, c \rangle \in C$ is satisfied if either $\phi(a) < \phi(b) < \phi(c)$ or $\phi(c) < \phi(b) < \phi(a)$ is true. The goal is to find an ordering ϕ such that all ordered triples in C are satisfied.*

Given an instance (L, C) of BETWEENNESS, we construct an instance, $G = (V, E)$, with edge partition, $P = \{\text{Red}, \text{Green}, \text{Blue}\}$, of UPBE-3 such that a subsequence of a solution π corresponds to a valid ordering ϕ of L in the BETWEENNESS instance. For each element $x \in L$, we create vertices x_1, \dots, x_{2m-1} in V . We call these vertices the *element vertices*. Our reduction uses two types of gadgets: *ordered triple gadgets* and *order preserving gadgets*. Their function is to enforce a betweenness constraint given by an element of C and to ensure that the order of element vertices of subscript j in π , with $j \in \{1, \dots, 2m-2\}$, is the reverse order of element vertices of subscript $j+1$, respectively. We prove that (G, P) admits an upward book embedding given by π if and only if the order of element vertices of the same subscript in π corresponds to a solution ϕ for the (L, C) BETWEENNESS instance.

3.1 Gadgets

Ordered Triple Gadget. We order the set C arbitrarily. For the $(\frac{i+1}{2})$ -th ordered triple $\langle a, b, c \rangle \in C$, where i is an odd integer between and including 1 and $2m-1$, we construct an *ordered triple gadget* that enforces the betweenness constraint on the triple of element vertices a_i, b_i, c_i in π . Specifically, we create the following nodes and edges.

Definition 2 (Ordered Triple Gadget). Let (L, C) be an instance of BETWEENNESS. Order the set C arbitrarily. For the $\binom{i+1}{2}$ -th ordered triple $\langle a, b, c \rangle$, where i is an odd integer between and including 1 and $2m - 1$, construct nodes $l_i, \alpha_i, \omega_i, a'_i, b'_i, c'_i$, and h_i . Then, create directed edges $(l_i, \alpha_i), (l_i, \omega_i) \in \text{Red}$, $(\alpha_i, a'_i), (\alpha_i, b'_i), (\omega_i, b'_i), (\omega_i, c'_i) \in \text{Blue}$, and $(a'_i, h_i), (b'_i, h_i), (c'_i, h_i) \in \text{Green}$.

Refer to Fig. 1 for an example construction. Nodes a'_i, b'_i and c'_i are respectively connected to a''_i, b''_i and c''_i by an edge in Red (where a''_i, b''_i , and c''_i are part of the order preserving gadget defined in Definition 3). The topologically earliest and latest nodes in the gadget are respectively l_i and h_i . The choice between $\pi(a'_i) < \pi(b'_i) < \pi(c'_i)$ and $\pi(a'_i) > \pi(b'_i) > \pi(c'_i)$ (and hence the choice between $\pi(a_i) < \pi(b_i) < \pi(c_i)$ and $\pi(a_i) > \pi(b_i) > \pi(c_i)$) is encoded in the choice between $\pi(\alpha_i) < \pi(\omega_i)$ and $\pi(\alpha_i) > \pi(\omega_i)$ as we prove in Lemmas 1 and 2.

Lemma 1. Given a positive instance (G, P) containing the ordered triple gadget shown in Fig. 1 (left), if $\pi(h_i) < \min(\pi(a''_i), \pi(b''_i), \pi(c''_i))$ then either $\pi(a''_i) < \pi(b''_i) < \pi(c''_i)$ or $\pi(c''_i) < \pi(b''_i) < \pi(a''_i)$.

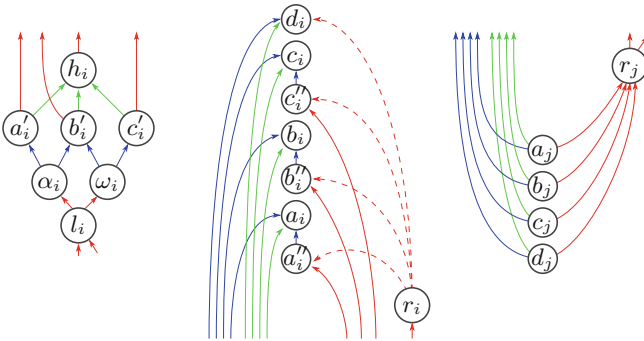


Fig. 1. Ordered triple gadget (left) and order preserving gadget for odd i (center) and for even j (right). The red edges from a'_i, b'_i and c'_i from the ordered triple gadget lead to a''_i, b''_i , and c''_i in the order preserving gadget. The red edge from h_i directs to r_i . (Color figure online)

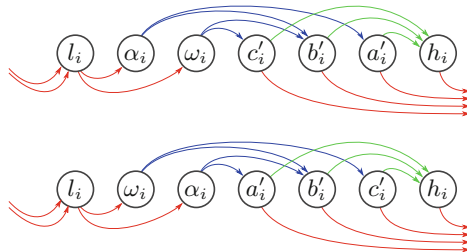


Fig. 2. The two possible embeddings of the ordered triple gadget. (Color figure online)

Proof. Notice that h_i must appear after a'_i, b'_i and c'_i due to topological order. By the assumption in the lemma, h_i be before a''_i, b''_i, c''_i respectively. We first prove that $\min(\pi(a'_i), \pi(b'_i), \pi(c'_i)) > \max(\pi(\alpha_i), \pi(\omega_i))$, i.e., no vertex in the gadget must occur between α_i and ω_i . Because of the topological order, $\pi(b'_i) > \max(\pi(\alpha_i), \pi(\omega_i))$. Suppose $\pi(\alpha_i) < \pi(\omega_i)$ (see Fig. 2 (top)). By topological order $\pi(c'_i) > \pi(\omega_i) = \max(\pi(\alpha_i), \pi(\omega_i))$. If $\pi(a'_i) < \pi(\omega_i)$, since $\pi(\omega_i) < \pi(a''_i)$, the red edges (a'_i, a''_i) and (h_i, ω_i) would intersect, a contradiction. The case when $\pi(\alpha_i) > \pi(\omega_i)$ is symmetric.

Trivially, either $\pi(\alpha_i) < \pi(\omega_i)$ or $\pi(\alpha_i) > \pi(\omega_i)$. We first assume $\pi(\alpha_i) < \pi(\omega_i)$ (Fig. 2 (top)). Then $\pi(c'_i) < \pi(b'_i) < \pi(a'_i)$, or else at least one pair of blue edges $((\alpha_i, b'_i), (\alpha_i, a'_i), (\omega_i, c'_i), (\omega_i, b'_i))$ would cross. Since all red edges $(a'_i, a''_i), (b'_i, b''_i)$ and (c'_i, c''_i) nest around h_i , every pair of such edges must be nested. Therefore, $\pi(a''_i) < \pi(b''_i) < \pi(c''_i)$. The case when $\pi(\alpha_i) > \pi(\omega_i)$ is symmetric. \square

Order Preserving Gadget. By Lemma 1, each ordered triple gadget enforces a betweenness constraint on vertices a''_i, b''_i and c''_i . The *order preserving gadgets* serve two purposes: ensuring that the i -th betweenness constraint is enforced in the i -th copy of element vertices; and ensuring that each copy of element vertices must occur in the reverse order of its predecessor. That implies that every other copy of element vertices occur in exactly the same order. We build $2m - 1$ order preserving gadgets, the j -th gadget containing x_j for each $x \in L$.

Definition 3 (Order Preserving Gadget). For each odd i in the range $[1, 2m - 1]$, we build the following nodes and edges:

1. Nodes $a''_i, b''_i, c''_i, x_i^p$ for $p \in [1, n]$, and r_i .
2. Directed edges $(a''_i, a_i), (b''_i, b_i), (c''_i, c_i) \in \text{Red}$.
3. Directed paths of length 7 connecting r_i to x_i^p for all $p \in [1, n]$ and where $x_i^p \neq a_i, b_i, c_i$. The paths alternate between red and green edges.
4. Directed paths of length 7 connecting r_i to a''_i, b''_i , and c''_i with alternating red and green edges.

For each even j in the range $[1, 2m - 1]$, we build the following nodes and edges:

1. Nodes x_j^p for $p \in [1, n]$ and r_j .
2. Directed edges $(x_j^p, r_j) \in \text{Red}$ for all $p \in [1, n]$.

The gadget is divided into two parts: the *elements part* containing the element vertices, and the *order preserving tree* whose root is labeled r_i or r_j . Figure 1 (center) shows an example of an order preserving gadget containing element vertices with odd subscript. Such instances are connected to ordered triple gadgets by three incoming red edges and have the vertex r_i as the lowest vertex in the topological order. The dashed edges represent a path of length 7 of alternating red/green edges that are connected to the element vertex x_i if $x \in L$ is not in the $(\frac{i+1}{2})$ -th ordered triple, or connected to the vertex x''_i otherwise. The vertices

x''_i for an element x in the i -th ordered triple are then connected to the element vertex x_i by a blue edge. For gadgets that contain element vertices with even subscript j (Fig. 1 (right)), r_j is the highest vertex in the topological order. For even $j \in \{2, \dots, 2m - 2\}$, we connect x_j to x_{j-1} with a blue edge and x_j to x_{j+1} with a green edge, for all $x \in L$ (see Fig. 6).

For odd i , the order preserving tree consists of n paths of length 7 of alternating red/green edges connected to the i -th element vertex (represented in Fig. 1 (center) as dashed arrows). Informally, their purpose is to allow such paths to “cross” the vertices connected to the i -th ordered triple gadget by red edges, while r_i as the first vertex in the topological order of the order preserving gadget.

Lemma 2. *Let (G, P) be a positive instance containing an order preserving gadget of odd index i connected to an ordered triple gadget representing $\langle a, b, c \rangle$. If there exists a set of blue edges (s_{i-1}, x_i) , $\pi(s_{i-1}) < \pi(r_i)$ and $\pi(s_{i-1}) < \min(\pi(a''_i, b''_i, c''_i))$, from some vertices s_{i-1} for all $x \in L$, then either $\pi(a_i) < \pi(b_i) < \pi(c_i)$ or $\pi(a_i) > \pi(b_i) > \pi(c_i)$.*

Proof. By topological order, $\pi(r_i) < \pi(x_i)$ for all $x \in L$. Then, all blue edges of the form (s_{i-1}, x_i) must nest around r_i . By Lemma 1, the order of vertices a''_i, b''_i and c''_i must obey the betweenness constraint $\langle a, b, c \rangle$. Since $\pi(y''_i) > \pi(r_i), y \in \{a, b, c\}$, if $\pi(a''_i) < \pi(b''_i)$ then $\pi(a_i) < \pi(b_i)$ or else edges (a''_i, a_i) and (s_{i-1}, b_i) would cross. With similar arguments, we can show that the order of the i -th element vertices must obey the betweenness constraint $\langle a, b, c \rangle$. \square

Lemma 3. *Let (G, P) be a positive instance containing three subsequent order preserving gadgets with indices $j - 1, j,$ and $j + 1$ where j is an even integer in $\{2, \dots, 2m - 2\}$. If $\pi(r_j) < \min\{\pi(r_{j-1}), \pi(r_{j+1})\}$, the element vertices with subscript $j+1$ appear in π in the same order as the element vertices with subscript $j - 1$.*

Proof. We prove that the order of the element vertices with subscript $j - 1$ is the reverse order of the element vertices with subscript j in π . By a similar argument, we can then show the same for j and $j + 1$, completing the proof. Notice that, since j is even, $\pi(x_j) < \pi(r_j), \pi(r_j) < \pi(r_{j-1}),$ and $\pi(r_{j-1}) < \pi(x_{j-1})$ for all $x \in L$ due to the topological order of the vertices. By the assumption in the lemma, all blue edges (x_j, x_{j-1}) nest around r_j and r_{j-1} . Therefore, any pair of such edges must be nested or they would cross. Hence, if $\pi(b_j) > \pi(a_j)$, then $\pi(b_{j-1}) < \pi(a_{j-1})$ or the blue edges (b_j, b_{j-1}) and (a_j, a_{j-1}) would cross (see Fig. 6 for example). Therefore, the order of the $(j - 1)$ -th copy of element vertices is the reverse order of the j -th copy. The same argument holds for x_j and x_{j+1} for all $x \in L$. Given that the order of the $(j - 1)$ -th and the order of the $(j + 1)$ -th copy of the element vertices are in reverse order of the j -th copy, the order of the $(j - 1)$ -th and $(j + 1)$ -th copies of the element vertices must be the same. \square

The next corollary immediately follows from Lemma 3.

Corollary 1. *If $\pi(r_j) < \min\{\pi(r_{j-1}), \pi(r_{j+1})\}$ for all even $j \in \{2, \dots, 2m-2\}$, then all element vertices with even subscript, $j \in \{2, \dots, 2m-2\}$ appear in the same order and all element vertices with odd subscript, $i \in \{1, \dots, 2m-1\}$ appear in the same order. Furthermore, all element vertices with even subscript, j , appear in the reverse order of all element vertices with odd subscript, i .*

3.2 Final Reduction

We create n ordered triple gadgets as defined in Definition 2 and $2m-1$ order preserving gadgets as defined in Definition 3, connecting via the following set of edges:

1. $(a'_i, a''_i), (b'_i, b''_i), (c_i, c''_i) \in \text{Red}$ for all odd $i \in [1, 2m-1]$.
2. $(r_j, l_{j-1}), (r_j, l_{j+1}) \in \text{Red}$ for all even $j \in [1, 2m-1]$.
3. $(s, x_n^p) \in \text{Blue}$ for all $p \in [1, n]$.
4. $(x_j^p, x_{j-1}^p) \in \text{Blue}$ for all $p \in [1, n]$ and even $j \in [1, 2m-1]$.
5. $(x_j^p, x_{j+1}^p) \in \text{Green}$ for all $p \in [1, n]$ and even $j \in [1, 2m-1]$.
6. $(h_i, r_i) \in \text{Red}$ for all odd $i \in [1, 2m-1]$.
7. $(x_j^p, r_j) \in \text{Red}$ for all $p \in [1, n]$ and even $j \in [1, 2m-1]$.
8. $(r_{2m-2}, s), (s, l_{2m-1}) \in \text{Blue}$.
9. $(r_{2m-2}, l_{2m-1}) \in \text{Red}$.

We connect the ordered triple and order preserving gadgets as described above, obtaining an instance (G, P) of UPBE- k . Using this reduction, we prove that UPBE- k is NP-complete for $k \geq 3$.

Theorem 1. *UPBE- k is NP-complete for $k \geq 3$.*

The proof follows from Lemmas 1, 2, and 3 and the constructions of the gadgets; please refer to our full paper for the proof [ADHL17].

4 UMPBE

In this section, we discuss the UPWARD MATCHING-PARTITIONED BOOK EMBEDDING problem, where given an instance (G, P) , each set of the partition P induces a subgraph in G that is a matching (i.e. no vertex is incident to more than one edge of each set of the partition). We first show UMPBE-4 is NP-complete and then show that UMPBE-2 is solved in $O(n)$ time. When $|P| = 1$, the algorithm in [HP99] for UPBE-1 can also solve UMPBE-1 in $O(n)$ time.

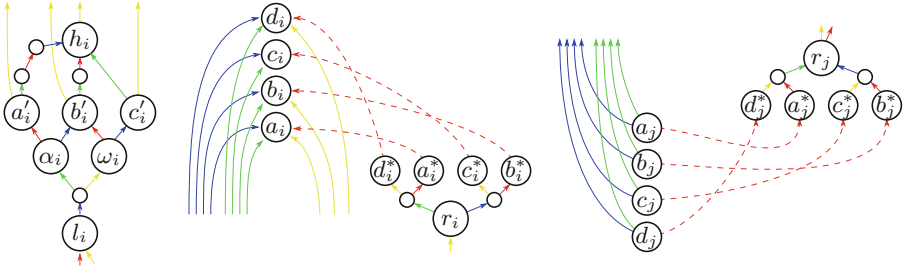


Fig. 3. Gadgets for the reduction to UMPBE-4. (Color figure online)

4.1 UMPBE-4

Theorem 2. UMPBE- k is NP-complete for $k \geq 4$.

Proof. As with UPBE, it is clear that UMPBE is in NP since an order π of vertices of G serves as a certificate. We show NP-hardness by reducing from BETWEENNESS, adapting the proof of Theorem 1 to UMPBE-4. We again refer to the partitions in $P = \{\text{Red, Blue, Green, Yellow}\}$ as colors. The gadgets adapted from Sect. 3.1 are shown in Fig. 3.

For odd i in $\{1, \dots, 2m - 1\}$ we connect gadgets with yellow edges (h_i, r_i) and (r_{i-1}, l_i) (if $i > 1$), and with the red edge (r_{i+1}, l_i) (if $i < 2m - 1$). Lemmas 1 holds for the new gadget replacing x''_i by x_i . We omit its proof due to the similarity. The dashed arrows in Fig. 3 represent paths of alternating colors as described in the next paragraph. Lemma 3 also trivially holds. Therefore, given a valid order π of vertices of G , the order of element vertices corresponds to a solution ϕ of the BETWEENNESS instance.

It remains to show that, given a solution ϕ of the BETWEENNESS instance, we can obtain a solution π for the produced instance. The order in which the gadgets are embedded are the same as in the proof of Theorem 1 and, therefore, no edge between gadgets cross. We now show that each gadget has a cross-free embedding using ϕ . The embedding of the ordered triple gadget is very similar to that shown in Fig. 2 and we chose $\pi(\alpha_i) > \pi(\omega_i)$ or $\pi(\alpha_i) < \pi(\omega_i)$ depending on whether a appears before c or vice-versa in ϕ . In the order preserving gadget, we use the same order (resp., reverse order) of ϕ for even (resp., odd) j .

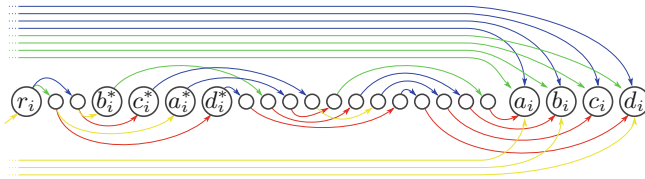


Fig. 4. Embedding the order preserving gadget in Fig. 3 (center). (Color figure online)

Notice that the order preserving gadget now contains a binary tree and we cannot choose arbitrarily the order of vertices x_j^* for $x \in L$. We call this tree the *binary order preversing tree* that ensures that $\pi(x_i) > \pi(r_i)$ and $\pi(x_j) < \pi(r_j)$ for all $x \in L$ for odd i and even j . The tree is obtained by ordering the vertices x_j^* arbitrarily and building a binary tree that alternates between green/blue and yellow/red edges in order for the induced subgraph of each color to be a matching. The paths connecting x_j^* to x_j allows us to order the element vertices x_j using ϕ , independent of the order of vertices x_j^* . We construct such paths in the following way. Each of the paths contains n edges. These paths alternate between green/blue and yellow/red edges starting with the opposite color group from the last row of the tree. Let x_j^t be the t -th vertex, $t \in \{1, \dots, n-1\}$, in an arbitrary order chosen as the order of the leaves of the order preserving tree. The colors of edges alternate between blue and red except for the t -th edge of the t -th vertex, such that the edge is green (resp., yellow) if the alternation would make it blue (resp., red). The embedding of the paths can be obtained by changing the order of the paths in an insertion sort manner, considering the last path (the path from d_j^* to d_j in Fig. 4) as the first element in the array and adding paths one by one in increasing order (see Fig. 4). Let x_j^t be the t -th vertex of the path from x_j^* to x_j and let y_j^t be the t -th vertex, $t \in \{1, \dots, n-1\}$, in the order chosen as the order of the leaves of the order preserving tree. Assume that the set A of all x_j^{t-1} is embedded so that the vertices are contiguous in the spine. We embed the set B of vertices of the form x_j^t immediately after the vertices in A in the reverse order in which they appear in A , apart from y_j^t . This order guarantees that no crossing is induced since all edges of the same color (x_j^{t-1}, x_j^t) are nested in parallel from A except for (y_j^{t-1}, y_j^t) which is of a different color. We can thus add y_j^t in any place in the ordering of B . \square

4.2 UMPBE-2

Theorem 3. *Given an instance $(G, \{E_1, E_2\})$ where $G = (V, E)$, $E = E_1 \dot{\cup} E_2$ and both (V, E_1) and (V, E_2) are matchings, UMPBE can be solved in $O(n)$ time where $n = |V|$.*

Proof. In positive instances, $G = (V, E)$ must be 2-page book embeddable and therefore planar [BK79]. Hence $|E| = O(n)$. Every connected component of the undirected version of G must be either a path or a cycle, or else the induced subgraph of the partitions would not be a matching. Furthermore, the edges connected in such paths or cycles must alternate in color. Each connected component can be solved separately since the concatenation of the solution (total order on vertices) of connected components is a solution of the original problem. Without loss of generality, we consider only the case when G is connected.

We provide a reduction to 1D origami when G is a path and a reduction to single vertex flat foldability if G is a cycle. The reduction runs in $O(n)$ time producing an instance with $n-1$ creases. Both the flat foldability of 1D origami and single vertex flat foldability can be determined in linear time [ABD+04, BH96]. A *face* in an 1D origami is defined as a segment in the 1D origami and a

crease is defined as a place where the origami can be folded. A *face* in a single vertex crease pattern is the space between creases. A 1D origami is defined by a line while a single vertex crease pattern is defined by a single vertex where rays originating from the vertex represent creases.

For both the case of the path and the cycle, we create an instance of 1D origami and single vertex flat foldability, respectively, in the following way. For each edge $e \in E$ we create a mountain crease if $e \in \text{Red}$ and a valley crease if $e \in \text{Blue}$. Each face of the produced instance represents a vertex in G . The reduction will thus produce an instance where each face of the origami has the same length, which can be viewed as a linkage formed by identical bars. If G is a path (resp., cycle), the output will be a list (resp., circular list) containing the assignment of the creases on a line segment (resp., a single vertex origami). Start with one endpoint of the path or with an arbitrary vertex of the cycle. Traverse the undirected version of G using BFS. For each edge traversed add *mountain* (resp., *valley*) to the end of the list if the traversed edge corresponds to an edge in E_1 with the same direction of the traversal or to an edge in E_2 in the opposite direction (resp., corresponds to an edge in E_2 with the same direction of the traversal or to an edge in E_1 in the opposite direction). Since every edge is traversed once, the size of the list is $n - 1$ (resp., n if a cycle). Thus, the only difference between single vertex crease patterns and 1D origami is that the faces form a cycle as opposed to a line segment, respectively.

Due to the similarity of the reduction models for paths and cycles, it suffices to show the equivalence between instances when G is a path. As previously stated, each face of the paper corresponds to a vertex in G . Each crease represents an edge in G and whether the crease is a mountain fold or a valley fold in the final state of the origami determines the partition of the edges of G into Red or Blue edges. If we consider the starting vertex as the leftmost face of the unfolded paper and that f_1 is not flipped in the folded state, a mountain fold puts the adjacent face f_2 below f_1 . Without loss of generality, the edge in G that represents the connection between f_1 and f_2 is in E_2 and points from f_2 to f_1 . By repeating the argument for every edge, we conclude that G represents the above/below relation of faces of the folded state of the 1D origami and E_1 (resp., E_2) represents the creases that lie right (resp., left) of the folded state (see Fig. 5). Then, it is easy to verify that the origami is flat-foldable iff $(G, \{E_1, E_2\})$ is a positive instance of UMPBE. \square

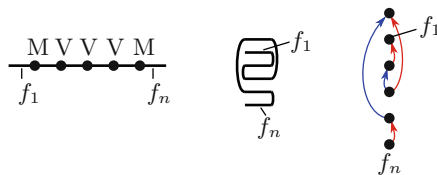


Fig. 5. A 1D origami crease pattern is shown (left) with mountain/valley labeled as M/V respectively, together with its folded state (center) and the corresponding UMPBE-2 instance (right). (Color figure online)

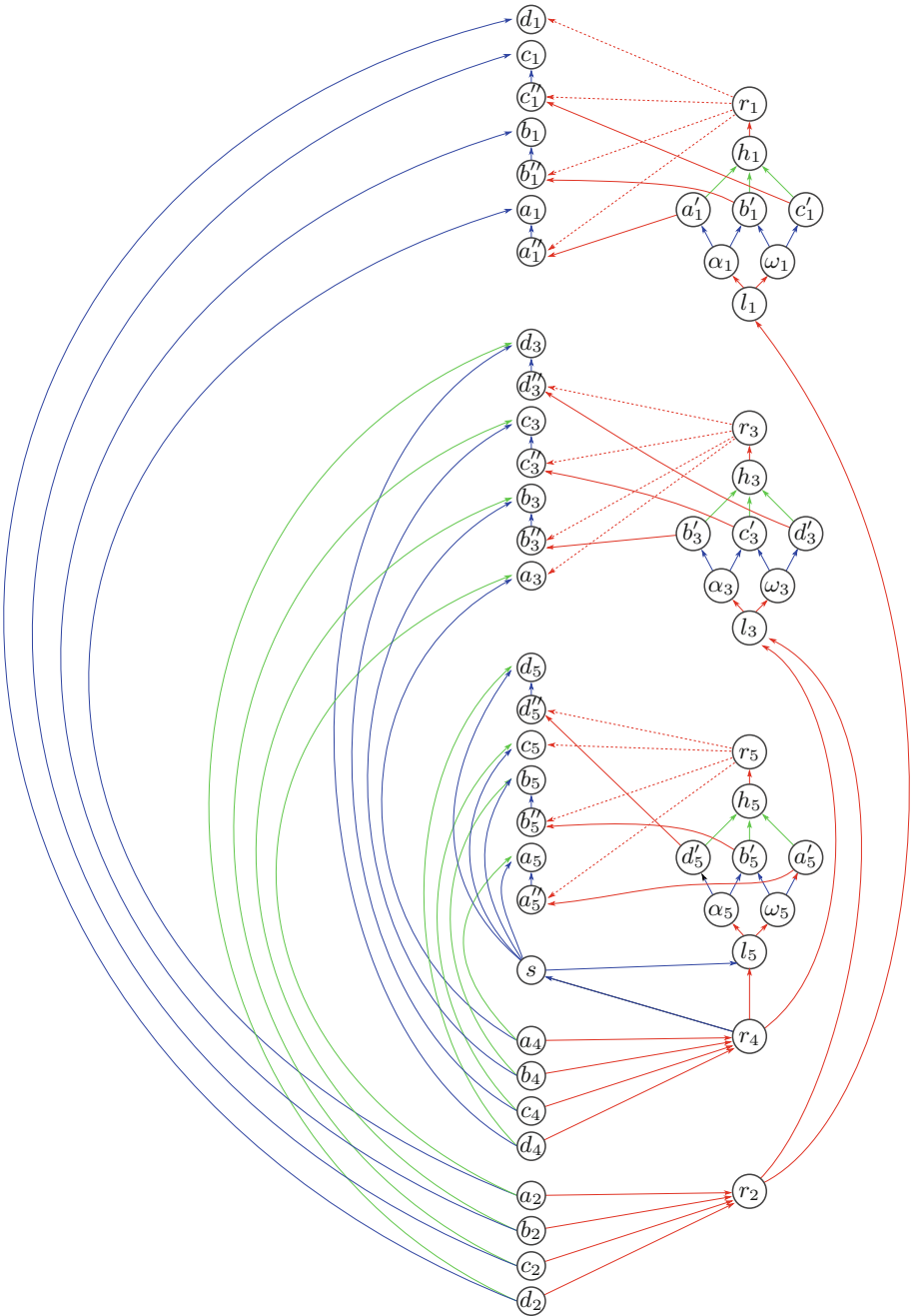


Fig. 6. Example of a full construction of a reduction. Here, the instance of BETWEENNESS is (L, C) where $L = \{a, b, c, d\}$ and $C = \{\langle a, b, c \rangle, \langle b, c, d \rangle, \langle d, b, a \rangle\}$. (Color figure online)

Acknowledgements. We thank Jack Edmonds for valuable discussions in August 1997 where he described how UPWARD MATCHING-PARTITIONED k -PAGE BOOK EMBEDDING generalizes the map folding problem. We also thank Therese Biedl for valuable discussions in 2007 about the complexity this problem.

This research was conducted during the 31st Bellairs Winter Workshop on Computational Geometry which took place in Holetown, Barbados on March 18–25, 2016. We thank the other participants of the workshop for helpful discussion and for providing a fun and stimulating environment. We also thank our anonymous referees for helpful suggestions in improving the clarity of our paper.

Supported in part by the NSF award CCF-1422311 and Science without Borders. Quanquan Liu is supported in part by NSF GRFP under Grant No. (1122374).

References

- [ABD+04] Arkin, E.M., Bender, M.A., Demaine, E.D., Demaine, M.L., Mitchell, J.S.B., Sethia, S., Skiena, S.S.: When can you fold a map? *Comput. Geom. Theor. Appl.* **29**(1), 23–46 (2004)
- [ABD12] Angelini, P., Di Bartolomeo, M., Di Battista, G.: Implementing a partitioned 2-page book embedding testing algorithm. In: Didimo, W., Patrignani, M. (eds.) *GD 2012*. LNCS, vol. 7704, pp. 79–89. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36763-2_8
- [ADHL17] Akitaya, H.A., Demaine, E.D., Hesterberg, A., Liu, Q.C.: Upward partitioned book embeddings. *CoRR*, abs/1708.06730 (2017)
- [ALN15] Angelini, P., Da Lozzo, G., Neuwirth, D.: Advancements on SEFE and partitioned book embedding problems. *Theoret. Comput. Sci.* **575**, 71–89 (2015)
- [BBKR15] Bekos, M.A., Bruckdorfer, T., Kaufmann, M., Raftopoulou, C.: 1-planar graphs have constant book thickness. In: Bansal, N., Finocchi, I. (eds.) *ESA 2015*. LNCS, vol. 9294, pp. 130–141. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48350-3_12
- [BGR16] Bekos, M.A., Gronemann, M., Raftopoulou, C.N.: Two-page book embeddings of 4-planar graphs. *Algorithmica* **75**(1), 158–185 (2016)
- [BH96] Bern, M., Hayes, B.: The complexity of flat origami. In: *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 1996*, pp. 175–183. Society for Industrial and Applied Mathematics, Philadelphia (1996)
- [BK79] Bernhart, F., Kainen, P.C.: The book thickness of a graph. *J. Comb. Theor. Ser. B* **27**(3), 320–331 (1979)
- [CLR87] Chung, F.R., Leighton, F.T., Rosenberg, A.L.: Embedding graphs in books: a layout problem with applications to VLSI design. *SIAM J. Algebraic Discrete Methods* **8**(1), 33–58 (1987)
- [DGDLW06] Di Giacomo, E., Didimo, W., Liotta, G., Wismath, S.K.: Book embeddability of series-parallel digraphs. *Algorithmica* **45**(4), 531–547 (2006)
- [DLM12] Demaine, E.D., Liu, E., Morgan, T.: A polynomial-time algorithm for $2 \times n$ map folding. Manuscript, 2012. See Tom Morgan’s M.Eng. thesis, “Map folding”, MIT (2012)
- [DW04] Dujmović, V., Wood, D.R.: On linear layouts of graphs. *Discrete Math’. Theor. Comput. Sci.* **6**(2), 339–358 (2004)

- [DW05] Dujmović, V., Wood, D.R.: Stacks, queues and tracks: layouts of graph subdivisions. *Discrete Math. Theor. Comput. Sci.* **7**(1), 155–202 (2005)
- [ENO97] Enomoto, H., Nakamigawa, T., Ota, K.: On the pagenumber of complete bipartite graphs. *J. Comb. Theor. Ser. B* **71**(1), 111–120 (1997)
- [FFR11] Frati, F., Fulek, R., Ruiz-Vargas, A.J.: On the page number of upward planar directed acyclic graphs. In: van Kreveld, M., Speckmann, B. (eds.) *GD 2011. LNCS*, vol. 7034, pp. 391–402. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-25878-7_37
- [HLR92] Heath, L.S., Leighton, F.T., Rosenberg, A.L.: Comparing queues and stacks as machines for laying out graphs. *SIAM J. Discrete Math.* **5**(3), 398–412 (1992)
- [HN14] Hong, S.-H., Nagamochi, H.: Simpler algorithms for testing two-page book embedding of partitioned graphs. In: Cai, Z., Zelikovsky, A., Bourgeois, A. (eds.) *COCOON 2014. LNCS*, vol. 8591, pp. 477–488. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08783-2_41
- [Hos12] Hoske, D.: Book embedding with fixed page assignments. Bachelor Thesis (2012)
- [HP97] Heath, L.S., Pemmaraju, S.V.: Stack and queue layouts of posets. *SIAM J. Discrete Math.* **10**(4), 599–625 (1997)
- [HP99] Heath, L.S., Pemmaraju, S.V.: Stack and queue layouts of directed acyclic graphs: Part II. *SIAM J. Comput.* **28**(5), 1588–1626 (1999)
- [HPT99] Heath, L.S., Pemmaraju, S.V., Trenk, A.N.: Stack and queue layouts of directed acyclic graphs: Part I. *SIAM J. Comput.* **28**(4), 1510–1539 (1999)
- [HR92] Heath, L.S., Rosenberg, A.L.: Laying out graphs using queues. *SIAM J. Comput.* **21**(5), 927–958 (1992)
- [MK16] McAdams, S., Kanno, J.: Oriented book embeddings. [arXiv:1602.02147](https://arxiv.org/abs/1602.02147) (2016)
- [Opa79] Opatrný, J.: Total ordering problem. *SIAM J. Comput.* **8**(1), 111–114 (1979)
- [Wie86] Wieggers, M.: Recognizing outerplanar graphs in linear time. In: Tinhofer, G., Schmidt, G. (eds.) *WG 1986. LNCS*, vol. 246, pp. 165–176. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-17218-1_57
- [Wig82] Wigderson, A.: The complexity of the hamiltonian circuit problem for maximal planar graphs. Technical report 298, EECS Department, Princeton University, Princeton, New Jersey (1982)
- [Yan89] Yannakakis, M.: Embedding planar graphs in four pages. *J. Comput. Syst. Sci.* **38**(1), 36–67 (1989)

Experimental Evaluation of Book Drawing Algorithms

Jonathan Klawitter¹(✉), Tamara Mchedlidze²(✉), and Martin Nöllenburg³(✉)

¹ University of Auckland, Auckland, New Zealand
jo.klawitter@gmail.com

² Karlsruhe Institute of Technology, Karlsruhe, Germany
mched@iti.uka.de

³ TU Wien, Vienna, Austria
noellenburg@ac.tuwien.ac.at

Abstract. A k -page book drawing of a graph $G = (V, E)$ consists of a linear ordering of its vertices along a *spine* and an assignment of each edge to one of the k *pages*, which are half-planes bounded by the spine. In a book drawing, two edges cross if and only if they are assigned to the same page and their vertices alternate along the spine. Crossing minimization in a k -page book drawing is NP-hard, yet book drawings have multiple applications in visualization and beyond. Therefore several heuristic book drawing algorithms exist, but there is no broader comparative study on their relative performance. In this paper, we propose a comprehensive benchmark set of challenging graph classes for book drawing algorithms and provide an extensive experimental study of the performance of existing book drawing algorithms.

1 Introduction

Book embeddings and book drawings are a fundamental and well-studied topic in graph theory and graph drawing. Combinatorially, a k -page book drawing of a graph $G = (V, E)$ consists of a cyclic linear ordering of its vertices along a *spine* and an assignment of each edge to one of the k *pages*, which are half-planes bounded by the spine. The spine and the k pages form a *book* (Fig. 1). Clearly, two edges $\{u, v\}$ and $\{w, z\}$ in a book drawing cross if and only if they are assigned to the same page and the four vertices alternate on the spine.

A book drawing is called a *book embedding* if it is crossing-free. The *book thickness* (or *pagenumber*) of a graph G is the smallest k such that G admits a k -page book embedding [5]. Deciding whether a graph can be embedded on k pages is an NP-complete problem even for $k = 2$ [6, 30] and there are many results about lower and upper bounds on the book thickness of specific graph classes. A long-standing open question [13] is to determine whether the book thickness of planar graphs is three or four. Yannakakis [41] showed that any planar graph can be embedded on four pages and there are planar graphs that

Preliminary results of this paper were presented in a poster at Graph Drawing 2016.

cannot be embedded on two pages. Likewise, the book thickness of k -planar graphs is open. Alam et al. [1] showed that there are 1-planar graphs that need four pages and that 16 is an upper bound.

If the number k of pages is given, a k -page book embedding may not exist. In this case, crossing minimization becomes the primary optimization goal. It reduces to two basic and interdependent combinatorial problems: the *vertex ordering* (VO) along the spine and the *page assignment* (PA) for the edges.

Again, computing the k -page book crossing number, i.e., the minimum number of crossings over all k -page book drawings of a graph, is an NP-hard problem [30,35] and fixed-parameter tractable algorithms for 1- and 2-page crossing minimization are known [2]. Book drawings are motivated by several applications, e.g., network visualization [4,16–18,39], VLSI design [40], RNA folding [11], and knot theory [14]. Various heuristic algorithms have been proposed in the literature. In addition, crossing minimization in book drawings has been the challenge problem of the Graph Drawing Contests in 2015 and 2016. Yet there are no broader comparative studies of these algorithms and no established set of challenging benchmark graph classes. In this paper, we introduce a comprehensive benchmark set for book drawing algorithms and provide the first extensive experimental study of the performance of state-of-the-art book drawing algorithms for multiple numbers of pages.

There are several heuristics for 2-page crossing minimization [8–10] with a fixed linear vertex ordering, as well as algorithms for the general 2-page crossing minimization problem [21,22]. Genetic and evolutionary crossing minimization algorithms have been proposed for one page [18], two pages [3,19,32], and any number of pages [34]. Further, neural networks have been used for 2-page crossing minimization [20,38] and for k -page crossing minimization [29].

Experimental evaluations have been performed by Satsangi et al. [34], who, however, excluded previously best performing algorithms by Baur and Brandes [4] and tested algorithms for VO and PA problems only independently from each other, not in combination. He et al. [23] performed an experimental study of several heuristics, but only for the 2-page crossing minimization problem.

Contributions and outline. In this paper, we determine the strengths and weaknesses as well as the relative performance of heuristic algorithms for the book drawing problem by means of a detailed quantitative experimental study. To this end, we present a list of state-of-the-art heuristic algorithms from the literature as well as some newly proposed heuristics in Sect. 2. Section 3 presents a collection of different graph classes together with suitable random graph generators to be used for creating benchmark graphs for our evaluation. Finally, Sect. 4 contains our comparative experimental evaluation. The main focus of our study is the relative performance of the heuristics in terms of crossing minimization depending on the properties of the benchmark instances, such as book thickness,

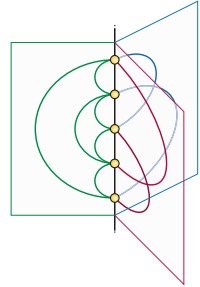


Fig. 1. 3-page book drawing of K_5 with two crossings.

graph size, edge density, and graph structure. Since our implementations are not optimized for a fast performance, we refrained from a detailed running time analysis. Some preliminary indications of the running times can be found in the appendix of the full version of this paper [27]. The code of our benchmark graph generators and of the book drawing algorithms can be found online¹.

2 Algorithms

We distinguished between constructive heuristics that have the common property that they consider each vertex and edge once, and local search heuristics that make several rounds re-considering the same vertices and edges iteratively. We evaluate these algorithms separately, as the latter can be seen as local search heuristics, which also use much more computation time. The constructive heuristics themselves can be characterized as VO heuristics, PA heuristics, and *combined* heuristics, that construct both VO and PA simultaneously.

2.1 Constructive Heuristics

Four of the heuristics presented in this section have not appeared in the literature earlier, namely `treeBFS`, `conGreedy`, `conGreedy+`, and `earDecomp` (see [26] for more details). Several additional heuristics are referenced, but not included in our study because they were always outperformed by the other presented heuristics in previous experimentation.

VO Heuristics. A VO heuristic considers vertices in some particular order and places them on the spine based on some criteria. An edge $\{u, v\}$ where only one of u and v (resp. both) has been placed on the spine is called *open* (resp. *closed*).

smallest degree DFS (sm1DgrDFS) [18]. DFS-based heuristics set the VO to be the order in which the vertices are visited by a depth-first traversal of the graph. The `sm1DgrDFS` heuristic starts with a smallest degree vertex and chooses a neighbor with smallest degree to proceed.

random DFS (randDFS) [3]. In contrast to `sm1DgrDFS`, `randDFS` starts with a random vertex and proceeds with a random neighbor.

tree-based BFS (treeBFS). This heuristic generates a breath-first spanning tree of the graph and embeds it crossing-free in a 1-page book yielding the VO. All three search based heuristics have a running time of $\mathcal{O}(m + n)$.

connectivity based (conCro) [4]. This heuristics builds the VO step by step. At each step it selects the vertex with the most neighbors already placed and breaks ties in favor of vertices with fewest unplaced neighbors (connectivity \rightarrow con). It places the vertex on that end of the already computed spine, where it introduces fewer crossings with open edges (crossings \rightarrow Cro). The intuition behind this heuristic is that the chosen vertex closes most open edges and opens fewest at ties. Its running time is $\mathcal{O}((m + n) \log n)$.

¹ Graph generators: github.com/joklawitter/GraphGenerators, book drawing algorithms: github.com/joklawitter/BookDrawingAlgorithms.

greedy connectivity based (conGreedy). Like `conCro` it selects the next vertex to place based on connectivity, however, it places it on any position (not just one of the end points) of the current spine where it introduces fewer crossings with closed edges. With $\mathcal{O}(m^2n)$ it has the highest running time.

Heuristics excluded, due to relatively poor performance, are among others a maximum neighborhood heuristic, a vertex-cover heuristic, a simple BFS heuristic [34], and variations of `conCro` [4].

PA Heuristics. The following first three heuristics share a general framework. They first compute an edge order according to some strategy and then place the edges one by one on the page where the increase in crossings is minimal.

ceil-floor (ceilFloor) [24]. In this strategy the edges are ordered non-increasingly by their length in a circular drawing.

length (eLen) [8,34]. Here the edges are ordered non-increasingly by the distance of their end vertices on a spine. Thus edge $\{1, n\}$ is listed first and any edge $\{i, i + 1\}$ last. Like `ceilFloor`, it has a $\mathcal{O}(m^2)$ running time.

circular (circ) [34]. The edges are enumerated in the order they are visited by the paths $P_1 \dots P_{\lceil \frac{n}{2} \rceil}$, where path P_i starts at vertex i and visits vertices $i + 1, i - 1, i + 2, \dots, i + \lceil \frac{n}{2} \rceil$. This heuristic is inspired by the fact that it achieves zero crossings for complete graphs on $\lceil \frac{n}{2} \rceil$ pages by placing edges of each path on a distinct page. It has a running time of $\mathcal{O}(n^4)$.

ear decomposition (earDecomp). Consider a circular drawing Γ_C of a graph $G = (V, E)$. The edge intersection graph is defined as $G_C = (E, \{\{e, e'\} \mid e, e' \in E \wedge e, e' \text{ cross in } \Gamma_C\})$. The heuristic `earDecomp` considers the circular drawing for the given VO, constructs its intersection graph G_C , and an ear decomposition of G_C , and then assigns the vertices of each ear (i.e., the edges of G) alternately to different pages. The intuition behind the heuristic is that it tries to put the conflicting edges to different pages. `earDecomp` can be implemented to run in $\mathcal{O}(m^2)$ time.

slope (slope) [22]. Consider the circular drawing with equally distributed vertices for a graph and VO. Then, the more the geometric slopes of two non-incident edges in this drawing differ, the more likely they cross. `slope` groups the edges based on their slopes and assigns each group to a page. It has a linear running time $\mathcal{O}(m)$.

Again, due to relatively poor performance, we excluded several other greedy variations [8,21,34], a dynamic programming and a divide and conquer approach [8].

Combined Heuristics. Almost all existing constructive heuristics compute a VO and a PA independently. He et al. [21] first combined the two problems. They extended `smlDgrDFS` such that whenever an edge is closed it is assigned to the page where it introduces the smallest number of crossings. We experimented with such extensions for `smlDgrDFS` and `randDFS` heuristics and concluded that they

performed worse than using them in combination with another PA heuristic [26]. The following heuristic utilizes this idea for `conGreedy`.

combined greedy connectivity based (`conGreedy+`). While constructing the VO like `conGreedy`, this heuristic considers the PA of already placed edges. More precisely, the best position for a new vertex is the position where this vertex's incident and now closed edges induce fewest new crossings. The PA is then accordingly extended to these newly closed edges. The heuristic's overall asymptotic running time is $\mathcal{O}(m^2n)$.

We note that the immediate page assignment done by `conGreedy+` also affects the computed VO. Hence, `conGreedy+` can also be used as VO heuristic by discarding the produced PA.

2.2 Local Search Heuristics

Local search heuristics take a given book drawing and try to reduce its number of crossings by performing local changes. Heuristics `greedyAlt` and `greedy+` are newly proposed, while `simAnn` has been proposed by Cibulka [7], who won the Automated Graph Drawing Challenge in 2015 [25].

alternating greedy search (`greedyAlt`). A single *vertex round* of this heuristic considers vertices in a random order, takes each of them in this order and places it on the position on the spine where it produces the least number of crossings. Here edges stay on the pages they are. A single *edge round* does the same with the edges: it considers edges in a random order and places them on the page where it produces the least number of crossings. `greedyAlt` alternates between vertex and edge rounds until it converges to a local minimum.

combined greedy search (`greedy+`). A single round of this heuristic is similar to `conGreedy+`, but the vertices are considered in a random order. Several rounds are performed until a local minimum is found.

simulated annealing (`simAnn`) [7]. This algorithm, depending on a temperature that decreases with each iteration, makes local changes to the book drawing and accepts them if they either improve the drawing, or with a certain probability depending on the temperature and how many crossings the move introduces. The moves in each iteration are (1) m times moving an edge to a random page, (2) $n\sqrt{n}$ times swapping a random vertex with its neighbor, (3) n times moving a vertex to a random position and greedily improving the assignment of its incident edges, and (4) $\frac{4}{n}$ times searches, in the fashion of `greedy+`, for the best position of a vertex. It runs 1000 iterations.

The literature contains similar greedy optimization algorithms for the VO [21, 36], another simulated annealing approach [34], evolutionary [24, 33], and neural network algorithms [29, 38]. However, they all were either restricted to only two pages or were outperformed by other heuristics in previous experiments [26].

3 Benchmark Graphs

Our previous experiments have shown that there is no fixed ranking for the performance of the construction heuristics in terms of crossing minimisation [26]. On the contrary, rankings depend on the number of pages, the edge densities of the graphs and their structuredness. We therefore selected nine different benchmark graph classes that vary in terms of density and structuredness and that are challenging for book drawing algorithms. This excludes some previously used graph classes such as trees or complete graphs. With our choices we aim to establish a set of benchmark graphs that will also serve as a basis for future investigations on book drawings.

Random. We use *random graphs* (Erdős-Rényi model) with linear density a , i.e., n -vertex graphs with an edges for $a = 2, \dots, 10$, and with quadratic density in terms of n , i.e., edge probabilities p .

Topological planar. To generate n -vertex *triangulated planar graphs*, we used a random edge-flip walk of length n^3 in the space of planar triangulations with a random Apollonian network as starting point. Known bounds suggest that n^3 is a suitable and still practical length [31].

Topological 1-planar. We generated 1-planar graphs by augmenting the 4-cycles in our planar triangulations with diagonals in a random order. This yielded on average 93% of the maximal number of edges in a 1-planar graph.

Geometric k -planar. For k from zero to four, we generated k -planar graphs as follows. Taking a random set of points in the plane, we sort them lexicographically, and then add an edge from a vertex (processed in sorted order) to an already processed vertex (in reversed order) only if the segment connecting them would not create more than k crossings in the current drawing. This process achieved on average 85% of the maximal number of edges.

k -tree. A k -tree is a recursively defined graph that is formed by starting with a k -clique and adding vertices and connecting them to all vertices of a k -clique of the current graph. We used this process to construct k -trees.

Hypercube. We used *hypercubes* Q_d of dimension d . They have $n = 2^d$ vertices and $m = \frac{1}{2}nd = \frac{1}{2}n \log n$ edges. Their book thickness is $d - 1$ [28].

Cube-connected cycles. A *cube-connected cycle* CCC_d of dimension d is a hypercube Q_d with vertices replaced by cycles of length d . They have $n = d2^d$ vertices and $m = 1.5n$ edges.

Toroidal mesh. A *toroidal mesh* $C_i \times C_j$ is the product graph of two cycles of length i and j . It has $n = ij$ vertices and $m = 2n$ edges.

3-toroidal mesh. A *3-toroidal mesh* $C_i \times C_j \times C_k$ is the product graph of three cycles of length i , j and k . It has $n = ijk$ vertices and $m = 3n$ edges.

The structuredness of these graph classes varies from very symmetric graphs, such as hypercubes and toroidal meshes (we call them *homogeneous*) to less homogeneous but still geometrically structured graphs, such as k -trees and k -planar graphs (we call them *structured*) to finally *random*, unstructured graphs.

4 Evaluation

In this section, we present the results of our experiments on the performance of the heuristic algorithms presented in Sect. 2 on the different benchmark graph classes introduced in Sect. 3. Our main focus in the evaluation is to analyze the relative performance of the book drawing heuristics, based on the density and structuredness of the graph classes, as well as the specified number of pages.

4.1 Experimental Setup

For each experiment, we used specific graphs, like hypercubes, 200 times or 200 graphs of the same class. For each graph, in the data representation, the order of the vertices and adjacency lists were randomized. The maximal number of pages considered in an experiment was either determined by the book thickness of a graph (if known), or limited to the first number where the best heuristic achieved less than ten crossings, or a 20 pages otherwise.

4.2 Constructive Heuristics

We first evaluate the constructive heuristics of Sect. 2.1 by considering all possible combinations of VO and PA heuristics. In previous experiments we observed that the right combination of them is crucial [26]. We thus refrained from testing them independently as done by Satsangi et al. [34]. By plotting the number of produced crossings for all the heuristic combinations and various graph classes, we observed that the parameters density, number of pages, and structure have a significant impact. To analyze how the performance depends on these three factors we consider each graph class individually, grouping them into homogeneous, structured and random graphs.

Homogeneous graphs.

Figure 2 shows the best heuristics on the hypercubes Q_4 to Q_9 and on different toroidal meshes. For Q_4 , having book thickness $d - 1 = 3$, conGreedy-ceil Floor could almost achieve its book embedding, however as the dimension increases the performance of the heuristics gets worse. The same holds for toroidal meshes and cube-connected cycles, which both have book thickness three [26, 37]. As we can observe by more detailed

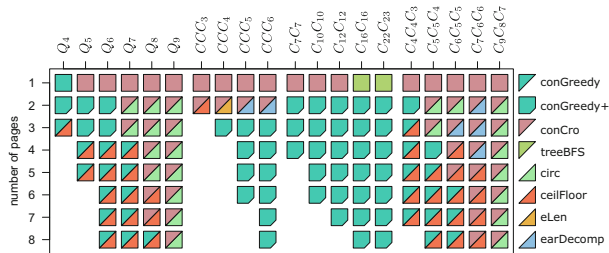
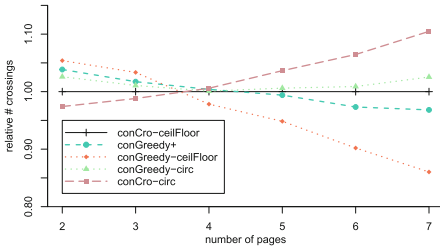


Fig. 2. Tile diagram for homogeneous graphs. One tile represents the heuristic or heuristic combination that achieved the best mean of crossings for the specific number of pages (row) and graph (column).

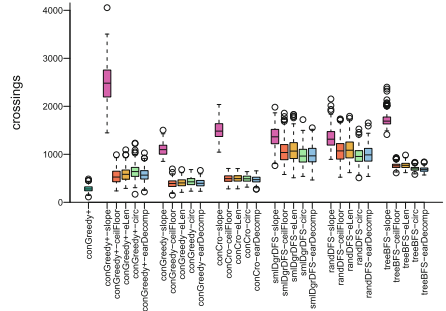
tile diagram, the performance of the heuristics gets worse. The same holds for toroidal meshes and cube-connected cycles, which both have book thickness three [26, 37]. As we can observe by more detailed

analysis (see the appendix of the full version of this paper [27]), all heuristics have on average more than a hundred crossings for hypercubes and toroidal meshes on book thickness many pages. For example, as shown in Fig. 3b, the best heuristics have on average more than 250 crossings for $C_{16}C_{16}$ on three pages. We suspect that the book thickness of 3-toroidal meshes is constant, and most likely below 8. If this holds, the performance of the heuristics is also poor for this graph class. The best performing heuristics (refer to Fig. 2) for hypercubes are `conGreedy-ceilFloor` and `conCro-circ`. For 3-toroidal meshes `conCro-ceilFloor` is also often the winner. For toroidal meshes and cube-connected cycles `conGreedy+` performs best.

Figure 3a illustrates the vertical dimension of the tile diagram for Q_7 . It shows the performance of heuristics depending on the number of pages. We see that the changes are smooth and that, however, for a high number of pages `conGreedy-ceilFloor` is substantially better than `conCro-circ` and the other heuristic. Figure 3b shows the results for all heuristics on $C_{16}C_{16}$ and three pages. Here `conGreedy+` performed best. Overall, we see that the choice of the VO heuristics has higher significance than PA, except if in combination with `slope`. It is also interesting that `treeBFS` performs significantly better than the other search based heuristics. Similar results appear for the other homogeneous graphs (see the full version of this paper [27] for figures).



(a) Hypercube Q_7 .



(b) $C_{16}C_{16}$, 3 pages.

Fig. 3. Number of crossings of the heuristics (a) with respect to `conCro-ceilFloor` (lower means less) for Q_7 depending on the number of pages, and (b) in absolute values for the toroidal mesh $C_{16}C_{16}$ and three pages.

Structured graphs. The structured graphs that we investigate are the k -planar graphs and k -trees. Figure 4 presents the best performing heuristics for geometric k -planar graphs and topological planar and 1-planar graphs. We observe that the difference in the structure of these graphs is crucial for the performance of the heuristics. For topological planar graphs `conGreedy-ceilFloor` dominates, while for geometric k -planar graphs `conCro-ceilFloor` is ahead in the majority of the cases. The PA heuristic `earDecomp` performs well for two pages.

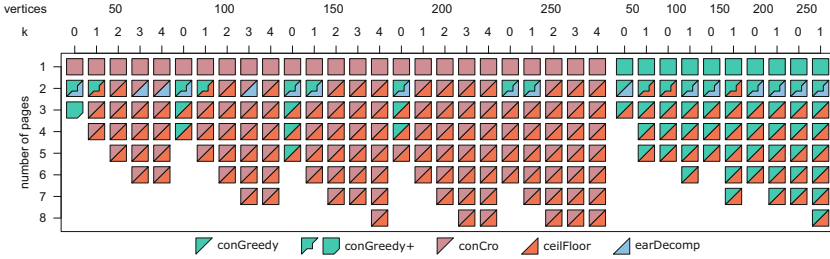
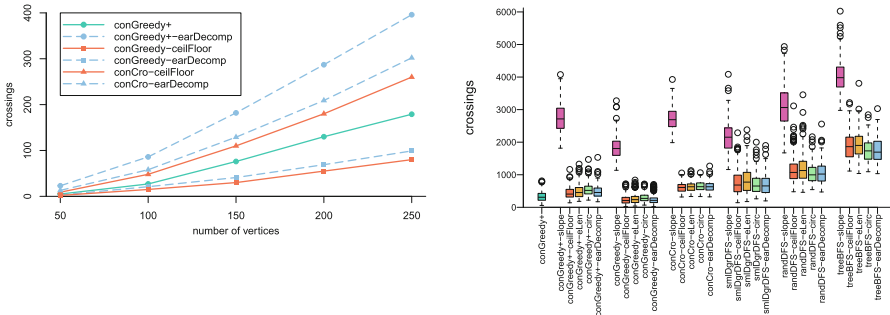


Fig. 4. Tile diagram for k -planar graphs, geometric (left) and topological (right).

Figure 5a shows the performance of the heuristics for topological planar graphs plotted as a function of the number of vertices for four pages, the upper bound for the book thickness of planar graphs [41]. The leading heuristic `conGreedy-ceilFloor` is not close to the optimum of zero, but achieves, for example, for graphs with 250 vertices on average 62 crossings. In contrast to the homogeneous graphs, we see in Fig. 5b, that the two DFS-based heuristics perform better on topological planar graphs, while `treeBFS` performs worst. Similar observations [27] can be made for k -planar graphs.



(a) Topological planar, 4 pages. (b) Topological planar, $n = 250$, 3 pages.

Fig. 5. Performance of the heuristics on topological planar graphs.

Figure 6 shows the overview of the results for k -trees. The diagram is dominated by `conGreedy-circ` and `conGreedy-ceilFloor`. We note that the book thickness of k -trees is at most $k + 1$ [13, 15] and then observe that `conGreedy+` achieves less than ten crossings on average for $k + 1$ pages. This becomes more apparent in Fig. 7, which shows the performance as a function of the number of pages for 8-trees with 250 vertices. We see that for a small number of pages, where `conGreedy-circ` dominates, the performance of all heuristics is comparable, while for more pages `conGreedy-ceilFloor` performs clearly better and finally, on nine pages, `conGreedy+` takes significant lead.

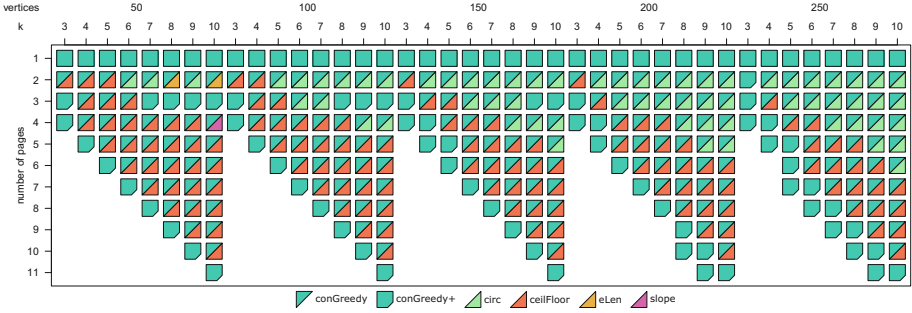


Fig. 6. Tile diagram for k -trees.

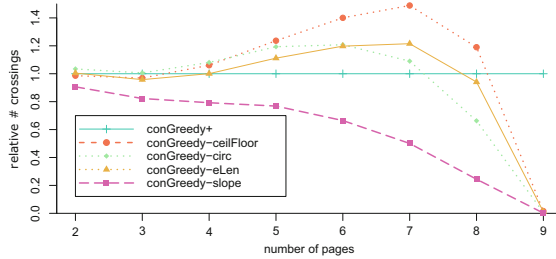


Fig. 7. Performance of conGreedy+ relative to conGreedy in combination with PA heuristics, for 8-trees, $n = 150$ and two to nine pages. A higher value means thus fewer crossings compared to conGreedy+.

Random graphs. The tile diagram in Fig. 8 for random graphs with linear number of edges shows again a clear pattern. Further investigation (see the full version of this paper [27]) shows that the transition between the best heuristics in Fig. 8 shifts smoothly along the number of vertices, pages and density. The heuristic conGreedy+ dominates for small number of pages and not too high density. However, if the number of pages is relatively high, we observe that conGreedy+ceilFloor and conGreedy-ceilFloor perform best. The differences between PA heuristics becomes more apparent for both higher density and more pages. The performance of slope gets significantly better with higher density, either with conGreedy or randDFS. The search based VO heuristics perform nearly equally, as do the greedy VO heuristics with conGreedy however slightly in the lead.

The good performance of slope seems natural, as with high density, and thus more edges per page, one edge with a slope different from other edges on the same page, is very likely to produce a lot of crossings. The results for random graphs with quadratic density illustrate this even further (see [27]). In fact, de Klerk et al. [12] conjecture that on complete graphs slope finds an optimal solution for any number of pages. They proved this for two and $\lfloor \frac{n}{2} \rfloor$ pages, with the latter being the book thickness of complete graphs K_n [5].

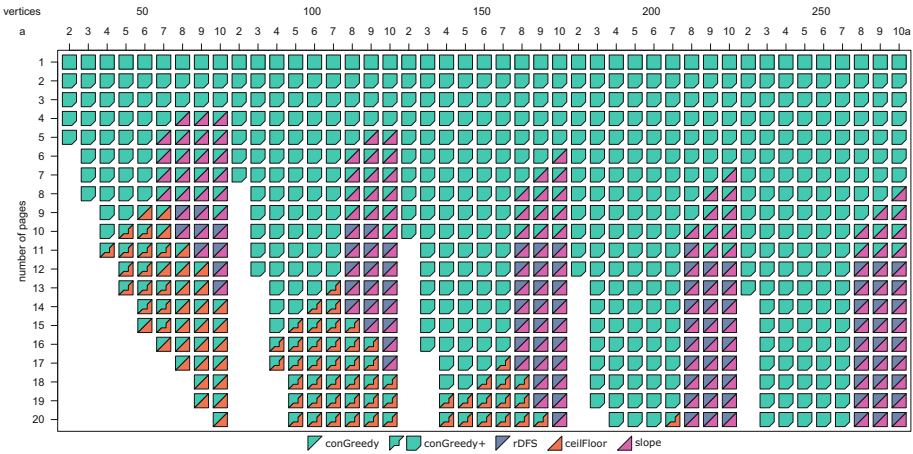


Fig. 8. Tile diagram for random graphs with linear density, i.e. roughly an edges.

4.3 Local Search Heuristics

In this section, we evaluate the local search heuristics **greedyAlt**, **greedy+** and **simAnn** (the core of the algorithm that won the 2015 Graph Drawing Contest), described in Sect. 2.2. Recall that **conGreedy+** is a combined constructive heuristic that considers VO and PA simultaneously, and as seen above often outperforms other heuristic combinations. With **greedy+**, we extended the idea of **conGreedy+** to a local search heuristic, which does multiple rounds on all vertices and edges, until a local minimum is found.

We tested the local search heuristics, similarly to the constructive heuristics, on graphs of different sizes, densities, structure, and with different numbers of pages (see Fig. 9 and Fig. 20 in the full version of this paper [27]). Here our findings are more clear-cut. In all our experiments **greedy+** performed best, followed

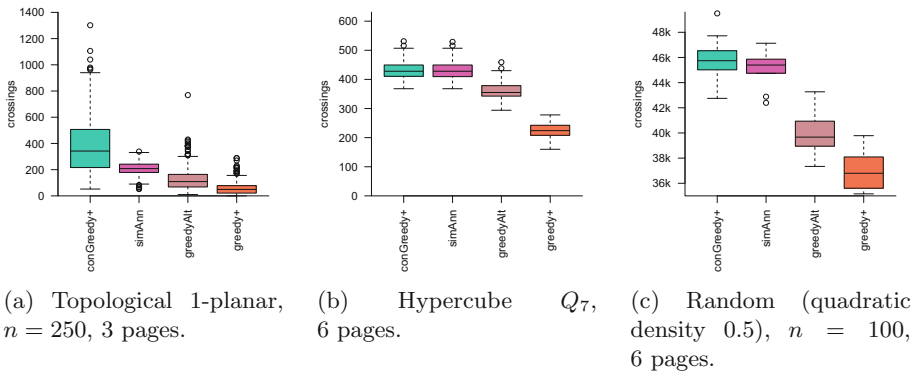


Fig. 9. Performance of the local search heuristics on graphs of various classes.

by **greedyAlt**. The heuristic **simAnn** had sometimes difficulties to improve the given book drawings, and performed worse than **greedyAlt**. The good performance of **greedy+** comes with a high trade-off in running time compared to **greedyAlt**. However, our implementation of **simAnn** was even slower².

5 Discussion and Conclusions

In our experiment, we investigated the relative performance of the heuristics presented in Sect. 2. We saw that the choice of the best constructive heuristic depends on several factors: density of the graphs, their structural properties and the number of pages. We could also see that the performance strongly depends on the selected combination of VO and PA heuristics.

We observed that constructive heuristics are mostly unable to achieve optimal results. For graph classes with known book thickness, the constructive heuristics could achieve very low crossing numbers only on k -trees. For homogeneous and structured graphs the results were far from optimal. We also observed that whenever the constructive heuristics performed poorly, the results of local search heuristics were also far from optimal. This fact, however, is not surprising, as even for trees, starting from random configuration, local search heuristics cannot achieve a book embedding [26]. Since most crossing numbers for the considered graph classes are unknown, we could not further investigate the relative performances in these cases.

The constructive heuristics **conGreedy** and its combined version **conGreedy+** performed best most of the times, but at a cost of higher running time. In several cases the VO heuristic **conCro** performed better than **conGreedy**, even though the former is just a restricted version of the latter. We observed that the **slope** heuristic performed well on dense graphs or in the case of a high ratio of the number of edges to the number of pages, which complies with the earlier conjecture about the power of **slope** to achieve optimal results for complete graphs. We also saw that our new VO heuristic **treeBFS** and PA heuristic **earDecomp** perform best or comparably to the other heuristics for homogeneous graphs and few pages. Regarding the local search heuristics, **greedy+** performed significantly better than the other local search heuristics on all graph classes, densities and number of pages. The simulated annealing algorithm performed even worse.

Our experiment can be extended into several directions that were beyond the limit of this paper: other theoretically and practically interesting graphs or graph classes, concentration on particular graph classes, more sophisticated implementations of the heuristics that would make it possible to test larger graphs, and finally a more detailed analysis of the results.

With respect to the tested heuristics, closer investigation would be necessary to understand why **conCro** performs better than **conGreedy** in several cases. It would also be of interest to see whether the performance of **treeBFS** on regular graphs could be further improved by derandomizing the way the BFS tree is

² We implemented our algorithms in Java and tested on a standard home computer (Intel[®] Core[™] i5-6600, 3.3 GHz, 8 GB RAM and Windows OS).

constructed. Concerning local search heuristics, an interesting open question is whether the optimisation of the VO or the PA is more influential on the overall performance of a heuristics.

References

1. Alam, M.J., Brandenburg, F.J., Kobourov, S.G.: On the book thickness of 1-planar graphs. CoRR, abs/1510.05891, October 2015. <http://arxiv.org/abs/1510.05891>
2. Bannister, M.J., Eppstein, D.: Crossing minimization for 1-page and 2-page drawings of graphs with bounded treewidth. In: Duncan, C., Symvonis, A. (eds.) GD 2014. LNCS, vol. 8871, pp. 210–221. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45803-7_18
3. Bansal, R., Srivastava, K., Varshney, K., Sharma, N., et al.: An evolutionary algorithm for the 2-page crossing number problem. In: Evolutionary Computation (CEC 2008), pp. 1095–1102. IEEE, June 2008. <https://doi.org/10.1109/CEC.2008.4630933>
4. Baur, M., Brandes, U.: Crossing reduction in circular layouts. In: Hromkovič, J., Nagl, M., Westfechtel, B. (eds.) WG 2004. LNCS, vol. 3353, pp. 332–343. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30559-0_28
5. Bernhart, F., Kainen, P.C.: The book thickness of a graph. J. Comb. Theor. Ser. B **27**(3), 320–331 (1979). [http://dx.doi.org/10.1016/0095-8956\(79\)90021-2](http://dx.doi.org/10.1016/0095-8956(79)90021-2)
6. Chung, F.R.K., Leighton, F.T., Rosenberg, A.L.: Embedding graphs in books: a layout problem with applications to vlsi design. SIAM J. Algebraic Discrete Methods **8**(1), 33–58 (1987). <http://dx.doi.org/10.1137/0608002>
7. Cibulka, J.: Simulated annealing book embedder (2015). <https://github.com/josefcibulka/book-embedder>
8. Cimikowski, R.: Algorithms for the fixed linear crossing number problem. Discrete Appl. Math. **122**(1), 93–115 (2002). [http://dx.doi.org/10.1016/S0166-218X\(01\)00314-6](http://dx.doi.org/10.1016/S0166-218X(01)00314-6)
9. Cimikowski, R.: An analysis of some linear graph layout heuristics. J. Heuristics **12**(3), 143–153 (2006). <http://dx.doi.org/10.1007/s10732-006-4294-9>
10. Cimikowski, R., Mumey, B.: Approximating the fixed linear crossing number. Discrete Appl. Math. **155**(17), 2202–2210 (2007). <http://dx.doi.org/10.1016/j.dam.2007.05.009>
11. Clote, P., Dobrev, S., Dotu, I., Kranakis, E., Krizanc, D., Urrutia, J.: On the page number of RNA secondary structures with pseudoknots. J. Math. Biol. **65**(6–7), 1337–1357 (2012). <http://dx.doi.org/10.1007/s00285-011-0493-6>
12. de Klerk, E., Pasechnik, D.V., Salazar, G.: Improved lower bounds on book crossing numbers of complete graphs. SIAM J. Discrete Math. **27**(2), 619–633 (2013). <http://dx.doi.org/10.1137/120886777>
13. Dujmović, V., Wood, D.R.: Graph treewidth and geometric thickness parameters. Discrete Comput. Geom. **37**(4), 641–670 (2007). <http://dx.doi.org/10.1007/s00454-007-1318-7>
14. Dynnikov, I.A.: Three-page approach to knot theory. Encoding and local moves. Funct. Anal. Appl. **33**(4), 260–269 (1999). <http://dx.doi.org/10.1007/BF02467109>
15. Ganley, J.L., Heath, L.S.: The pagenumber of k -trees is $O(k)$. Discrete Appl. Math. **109**(3), 215–221 (2001). [http://dx.doi.org/10.1016/S0166-218X\(00\)00178-5](http://dx.doi.org/10.1016/S0166-218X(00)00178-5)
16. Gansner, E.R., Koren, Y.: Improved circular layouts. In: Kaufmann, M., Wagner, D. (eds.) GD 2006. LNCS, vol. 4372, pp. 386–398. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70904-6_37

17. Giacomo, E.D., Didimo, W., Liotta, G., Wismath, S.K.: Book embeddability of series-parallel digraphs. *Algorithmica* **45**(4), 531–547 (2006). <http://dx.doi.org/10.1007/s00453-005-1185-7>
18. He, H., Sýkora, O.: New circular drawing algorithms. In: Workshop on Information Technologies - Applications and Theory (ITAT 2004) (2004). <https://dspace.lboro.ac.uk/2134/2386>
19. He, H., Sýkora, O., Mäkinen, E.: Genetic algorithms for the 2-page book drawing problem of graphs. *J. Heuristics* **13**(1), 77–93 (2007). <http://dx.doi.org/10.1007/s10732-006-9000-4>
20. He, H., Sýkora, O., Mäkinen, E.: An improved neural network model for the two-page crossing number problem. *IEEE Trans. Neural Netw.* **17**(6), 1642–1646 (2006). <http://dx.doi.org/10.1109/TNN.2006.881486>
21. He, H., Sýkora, O., Salagean, A., Vrt'o, I.: Heuristic crossing minimisation algorithms for the two-page drawing problem. Technical report, Loughborough University (2006). <https://dspace.lboro.ac.uk/2134/2377>
22. He, H., Sýkora, O., Vrt'o, I.: Crossing minimisation heuristics for 2-page drawings. *Electron. Notes Discrete Math.* **22**, 527–534 (2005). <http://dx.doi.org/10.1016/j.endm.2005.06.088>
23. He, H., Sălăgean, A., Mäkinen, E., Vrt'o, I.: Various heuristic algorithms to minimise the two-page crossing numbers of graphs. *Open Comput. Sci.* **5**(1), 22–40 (2015). <https://doi.org/10.1515/comp-2015-0004>
24. Kapoor, N., Russell, M., Stojmenovic, I., Zomaya, A.Y.: A genetic algorithm for finding the pagenumber of interconnection networks. *J. Parallel Distrib. Comput.* **62**(2), 267–283 (2002). <http://dx.doi.org/10.1006/jpdc.2001.1789>
25. Kindermann, P., Löffler, M., Nachmanson, L., Rutter, I.: Graph drawing contest report. In: Di Giacomo, E., Lubiw, A. (eds.) GD 2015. LNCS, vol. 9411, pp. 531–537. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27261-0_43
26. Klawitter, J.: Algorithms for crossing minimisation in book drawings. Master's thesis, Karlsruhe Institute of Technology (2016)
27. Klawitter, J., Mchedlidze, T., Nöllenburg, M.: Experimental evaluation of book drawing algorithms. CoRR, abs/1708.09221, August 2017. <http://arxiv.org/abs/1708.09221>
28. Konoe, M., Hagihara, K., Tokura, N.: Page-number of hypercubes and cube-connected cycles. *Syst. Comput. Jpn.* **20**(4), 34–47 (1989). <http://dx.doi.org/10.1002/scj.4690200404>
29. López-Rodríguez, D., Mérida-Casermeyro, E., Ortíz-de-Lazcano-Lobato, J.M., Galán-Marín, G.: K -Pages graph drawing with multivalued neural networks. In: de Sá, J.M., Alexandre, L.A., Duch, W., Mandic, D. (eds.) ICANN 2007. LNCS, vol. 4669, pp. 816–825. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74695-9_84
30. Masuda, S., Nakajima, K., Kashiwabara, T., Fujisawa, T.: Crossing minimization in linear embeddings of graphs. *IEEE Trans. Comput.* **39**(1), 124–127 (1990). <http://dx.doi.org/10.1109/12.46286>
31. McShine, L., Tetali, P.: On the mixing time of the triangulation walk and other catalan structures. *Randomization Methods Algorithm Des.* **43**, 147–160 (1999)
32. Poranen, T., Mäkinen, E., He, H.: A simulated annealing algorithm for the 2-page crossing number problem. In: Proceedings of International Network Optimization Conference (INOC) (2007)

33. Satsangi, D., Srivastava K., Gursaran: A hybrid evolutionary algorithm for the page number minimization problem. In: Nagamalai, D., Renault, E., Dhanuskodi, M. (eds.) Trends in Computer Science, Engineering and Information Technology. Communications in Computer and Information Science, vol. 204. pp. 463–475. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24043-0_47
34. Satsangi, D., Srivastava, K., Srivastava, G.: K-page crossing number minimization problem: an evaluation of heuristics and its solution using gesakp. Memetic Comput. **5**(4), 255–274 (2013). <http://dx.doi.org/10.1007/s12293-013-0115-5>
35. Shahrokhi, F., Székely, L.A., Sýkora, O., Vrt'o, I.: The book crossing number of a graph. J. Graph Theor. **21**(4), 413–424 (1996). [https://doi.org/10.1002/\(SICI\)1097-0118\(199604\)21:4<413::AID-JGT7>3.0.CO;2-S](https://doi.org/10.1002/(SICI)1097-0118(199604)21:4<413::AID-JGT7>3.0.CO;2-S)
36. Six, J.M., Tollis, I.G.: A framework and algorithms for circular drawings of graphs. J. Discrete Algorithms **4**(1), 25–50 (2006). <http://dx.doi.org/10.1016/j.jda.2005.01.009>
37. Tanaka, Y., Shibata, Y.: On the pagenumber of the cube-connected cycles. Math. Comput. Sci. **3**(1), 109–117 (2010). <http://dx.doi.org/10.1007/s11786-009-0012-y>
38. Wang, J.: Hopfield neural network based on estimation of distribution for two-page crossing number problem. IEEE Trans. Circ. Syst. II Express Briefs **55**(8), 797–801 (2008). <http://dx.doi.org/10.1109/TCSII.2008.922373>
39. Wattenberg, M.: Arc diagrams: visualizing structure in strings. In: Information Visualization (INFOVIS 2002), pp. 110–116. IEEE (2002)
40. Yannakakis, M.: Linear and book embeddings of graphs. In: Makedon, F., Mehlhorn, K., Papatheodorou, T., Spirakis, P. (eds.) AWOC 1986. LNCS, vol. 227, pp. 226–235. Springer, Heidelberg (1986). https://doi.org/10.1007/3-540-16766-8_20
41. Yannakakis, M.: Embedding planar graphs in four pages. J. Comput. Syst. Sci. **38**(1), 36–67 (1989). [http://dx.doi.org/10.1016/0022-0000\(89\)90032-9](http://dx.doi.org/10.1016/0022-0000(89)90032-9)

Evaluations

Visual Similarity Perception of Directed Acyclic Graphs: A Study on Influencing Factors

K. Ballweg¹(✉), M. Pohl², G. Wallner², and T. von Landesberger¹

¹ Technische Universität Darmstadt, Darmstadt, Germany
{kathrin.ballweg,tatiana.von.landesberger}@gris.tu-darmstadt.de

² Vienna University of Technology, Vienna, Austria
margit@igw.tuwien.ac.at, guenter.wallner@tuwien.ac.at

Abstract. While visual comparison of directed acyclic graphs (DAGs) is commonly encountered in various disciplines (e.g., finance, biology), knowledge about humans' perception of graph similarity is currently quite limited. By graph similarity perception we mean how humans perceive commonalities and differences in graphs and herewith come to a similarity judgment. As a step toward filling this gap the study reported in this paper strives to identify factors which influence the similarity perception of DAGs. In particular, we conducted a card-sorting study employing a qualitative and quantitative analysis approach to identify (1) groups of DAGs that are perceived as similar by the participants and (2) the reasons behind their choice of groups. Our results suggest that similarity is mainly influenced by the number of levels, the number of nodes on a level, and the overall shape of the graph.

Keywords: Graphs · Perception · Similarity · Comparison
Visualization

1 Introduction

The visual comparison of directed acyclic graphs (DAGs) is a task encountered in various disciplines, e.g., in finance, biology, natural language processing, or social network analysis. The task is strongly influenced by the human perception of similarity since comparison builds upon making similarity judgments. In spite of the numerous occurrences of this task and recent papers surveying visual graph comparison techniques and visualizations [4, 12], knowledge about the human perception of graph similarity – especially for DAGs – is quite limited.

Only a few investigations address the comparison of graphs. Gleicher et al. [12] identify the basic types of techniques for visual comparison (juxtaposition, superposition, and explicit encoding). Tominski et al. [43] explicitly deal with the comparison of large node-link diagrams in superposition. They argue that interaction is essential for this process. Some interesting insights can be gained from the literature on dynamic graphs showing the evolution of node-link diagrams over time. The survey of Beck et al. [4] about visualizations of dynamic graphs

provides an overview of visualization options transferable to general visual graph comparison since dynamic graph visualization has an inherent comparison component. Others discuss the extension of these visualizations and techniques with features like highlighting of commonalities and differences or the effectiveness of difference maps [1–3, 5, 16]. However, none of these papers deal with the issue of similarity perception within the context of graph comparison. There is also a large amount of research on graph readability. This research is partially relevant since the DAGs need to be read in order to compare them. Examples include studies on edge crossings and mental map perseverance (e.g., [23, 35, 37, 39]). These aspects are, however, not in the focus of our attention. The research investigating the comparison of visualizations in general is also interesting. Pandey et al. [33] conducted an experiment to study the similarity perception of scatterplots. So, their work inspired our methodology.

To the best of our knowledge, there is no research focusing on how humans perceive the similarity of DAGs. We are especially interested in the *factors which influence the perception of similarity* (possibly, number of nodes/edges, edge crossings, etc.). We deem the knowledge about the influencing factors important for the generation of future actionable guidelines for comparative visualizations. Towards this end, we conducted a study with small, unlabeled synthetic DAGs and used card sorting as our methodology. We decided for these DAGs in order to be able to keep the number of to be tested factors manageable. However, because of our systematic procedure the study scope can be easily extended in the future. The DAGs are represented as node-link diagrams. We address two research questions: (1) *Which groups do the participants form?*, and (2) *Which factors did the participants consider to judge the similarity?*

Our results indicate that similarity is mainly influenced by the number of levels and the number of nodes on a level as well as the overall shape. We provide additional material - i.a. our study material, our collected data and our analysis results here:

www.gris.tu-darmstadt.de/research/vissearch/projects/DAGSimilarityPerception.

2 Related Work

While there exists an extensive body of research in perceptual psychology and pattern recognition on similarity judgments and dissimilarity measures (cf. e.g., [13, 34] for an overview), we will concentrate on work dealing with graphs and other types of plots.

Starting with graph visualization techniques and visual comparison techniques there exist several recent surveys of these areas (e.g., [4, 12, 15, 25, 44]). The basic techniques, that is, juxtaposition, superposition, and explicit encoding – following Gleicher et al.’s [12] classification – are sometimes enriched by emphasizing the correspondences between graphs [7, 16], e.g., by highlighting similar parts [3, 5, 16], or by emphasizing differences by collapsing the identical parts [1]. The enrichment, that is, emphasizing the commonalities or differences, usually relies on a similarity function respecting specific criteria. In this respect, Gao et al. [10] provide an overview of research done on graph edit distances, a

mathematical way to measure the similarity between pairwise graphs. However, it is still unknown whether the criteria on which existing similarity functions are based correspond to the criteria used by humans when visually comparing two or more node-link diagrams. Tominski et al. [43] proposed interaction techniques which aid users in doing comparison tasks and which were inspired by the real-world behavior of people when comparing information printed on paper. Getting a better understanding of the perceived differences and commonalities is likely to result in better visualization and interaction techniques.

Moreover, the existing body of work dealing with perceptual and cognitive aspects focuses primarily on the readability of single graphs. Several factors, including graph aesthetics (edge crossings [23, 37, 38], layout [8, 20, 29, 30, 32], graph design [17, 40], and graph semantics or content knowledge [24, 32, 39]) have been identified to be important for graph readability. Huang et al. [18] – concerned with sociograms – note that good readability is not enough to effectively communicate network structures, emphasizing that the spatial arrangement of the nodes also influences viewers in perceiving the structure of social networks.

While perceptual aspects of single graphs have been thoroughly investigated, literature dealing with perceptual aspects when comparing node-link visualizations is considerably more scarce. Notable papers in this space are the work of Bach et al. [3] and Ghani et al. [11] who are both concerned with dynamic graphs (cf. Beck et al. [4] for an overview). Noteworthy is also the work of Archambault et al. [2] who evaluated the effectiveness of difference maps which show changes between time slices of dynamic graphs. While we are not necessarily concerned with dynamic graphs these works are nonetheless relevant in our context as dynamic graphs are often analyzed by using discrete time-slices. In our own previous work [27] we provided an overview of methodological challenges when dealing with the investigation of graph comparison and described a first preliminary study targeted towards identifying factors which influence the recognition of graph differences in very small star-shaped node-link diagrams. The work presented in this paper can be viewed as a continuation of these efforts.

Looking beyond the perception of node-link diagrams literature is currently also quite limited when it comes to similarity perception of other types of plots, a sentiment shared by Pandey et al. [33] who investigated how human observers judge the similarity of scatterplots. Our quantitative analysis as presented in this paper is partly based on the methodology put forward by Pandey et al. Fuchs et al. [9] looked into how contours affect the recognition of data similarity in star glyphs. Likewise, Klippel et al. [22] investigated the similarity judgments of star glyphs using a methodology comparable to the one used by Pandey et al. [33] and us: Participants were shown various plots which they then had to group according to their perceived similarity.

3 Study Methodology

In this section, we present our study methodology. As noted above, Pandey et al.'s [33] work about the human similarity perception of large sets of scatterplots

strongly inspired our basic methodology, since we share the research questions for different data types. Furthermore, Pandey et al. substantiate that the methodological principle of card sorting produces valuable results for this type of research questions. For advantages, drawbacks, and the suitability of card sorting for research questions like ours see Sect. 3.4.

3.1 Research Questions

Our superordinate research question (RQ) is: *What factors influence human similarity perception of DAGs?* We firstly have to know the factors influencing the similarity judgment. Once we know the influencing factors, we can, for instance, research the specific degree of influence of a single factor as well as the interplay between the factors. In order to analyze our superordinate RQ, we formulate two subordinate ones:

- RQ1: *Which groups do the participants form?*
- RQ2: *Which factors did the participants consider to judge the similarity?*




3.2 Dataset

Creating an appropriate study dataset is challenging due to the large number of possible variations [27]. Therefore, we were forced to limit the number of DAGs. Our object of study were 69 small (6–9 nodes), unlabeled, synthetic DAGs visualized as node-link diagrams. In the following we will use the term DAG to also refer to its embedding. When creating the DAGs we considered *known factors influencing graph readability* (e.g., edge crossings) and *characteristics of DAGs from real-world datasets* (e.g., a node may be the child of more than one parent node). We decided to have synthetic and small DAGs in order to be able to keep the number of to be tested factors manageable and to evaluate them systematically. Because of our study and data creation methodology it is easily feasible to extend our results with further studies considering further factors. Especially since knowledge about human graph similarity perception is currently quite limited, we consider the manageability of the problem as crucial. The size of our DAGs is also realistic. They are comparable to cascades in finance and biology (e.g., [26, 28]) and directed acyclic word graphs [41].

We deem *factors of graph readability* as potentially important for visual graph comparison since in order to be able to visually compare DAGs it is necessary to read them. We consider *properties of real DAGs* as important for our studies since they influence the visual appearance of the DAGs. More importantly, considering properties of real DAGs strengthens the *realism* of our synthetic data and, consequently, the *transferability* of our results to real world use cases.

To create our dataset, we started with G_0 , as depicted in Fig. 1. G_0 is symmetric since it is easier to break symmetry than to introduce symmetry. We herewith cover *symmetric* and *asymmetric* DAGs on the basis of G_0 . This is important since humans are prone to symmetry [45]. G_0 is single-rooted since

this is typical for various real-world DAG datasets; e.g., cascades. To test *node* and *edge changes* (*addition of node(s) and edge(s)*) we had a two stage DAG creation process. First, we created the base graphs $G1 - G6$ and their reflections by adding one, two, and three nodes. We ensured that the addition of the node(s) is done in the *inner as well as the outer areas of $G0$* (cf. Fig. 1 - Base graphs). Secondly, we created all possible DAGs resulting from adding one and two edges (cf. Fig. 1 - Alternatives) using our custom-made *GraphCreator* – a tool to create all possible DAGs resulting from a specific change of a DAG, e.g., adding one edge to $G1 - G6$ and their reflections. Herewith, we ensure that we have the maximal possible variation from which we then sample our study dataset.

Down-sampling is necessary since the visual comparison of DAGs is a quite cognitive demanding task for the participants. For the down-sampling we considered the following factors: *edge crossing*, *visual layout*, *more than one parent node has the same child node*, and *long connections – typically across more than one level*. We considered *edge crossing* since it is a prominent factor in graph readability [23,37] for which reason we assume that it also plays a role in visual graph comparison. Furthermore, we considered the *visual layout* – another known factor from graph readability. The visual layout of DAGs does not contain any analytically relevant information about the DAGs’ data structure and properties. However, it still has a significant impact on graph readability which is why we deem it important to test its influence on visual graph comparison [8,18]. We test this factor by horizontally *reflecting $G2, G4$, and $G6$* (cf. Fig. 1). The ability to test the impact of *isomorphism* is a beneficial byproduct of this decision. We decided for a traditional hierarchical node-link diagram layout () with the root placed on top since Burch et al. found that this layout type outperforms other types such as upward layouts. In order to avoid confounding effects by destroying the mental map we did not optimize the layout after a DAG change; e.g., resolving an avoidable edge crossing. By visually inspecting DAGs from real-world datasets we found that two frequently occurring properties are: *more than one parent node has the same child node* ( – e.g., cascades in finance and biology), *long connections – typically across more than one level* ( – e.g., directed acyclic word graphs in natural language processing). Consequently, we considered them as factors for our final dataset. We did the down-sampling under the constraint of preserving the systematic variation (cf. Fig. 1 – Final dataset) and by ensuring that changes take place at the *inner as well as the outer areas of the respective base graph*. Our dataset is available here: website (<http://www.gris.tu-darmstadt.de/research/vissearch/projects/DAGSimilarityPerception/index.html>).

3.3 Participants

We recruited 20 volunteers (13 male, 7 female, between 20 and 60 years). We had no prerequisite of having experience with DAGs. This way our results are not limited to experienced users. In our opinion, it is more likely that experienced users know which factors really bear relevant information for the comparison task whereas for inexperienced users misconceptions are more likely. We are

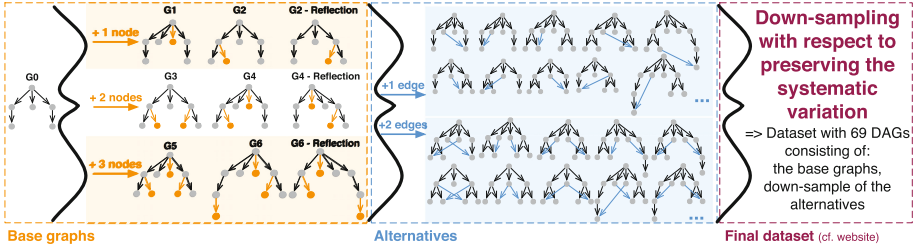


Fig. 1. Dataset creation: (I) Base graph creation by adding 1, 2, and 3 nodes to G_0 ensuring that the added node is placed at the *inner as well as the outer areas* of G_0 , (II) creation of all possible alternatives by adding one and two edges to the base graphs, (III) down-sampling of the alternatives considering the factors described in Sect. 3.2. (Color figure online)

convinced that if we want to understand the human similarity perception and as a consequence improve comparative visualizations, we need a varying range of expertise with DAGs. Our participants had a diverse educational level (vocational training, undergraduate, graduate, post-graduate) and came from various disciplines. Two of our participants had basic knowledge in information visualization and five had advanced knowledge. In spite of this, the participants’ experience with DAGs varied vastly.

3.4 Study Procedure

Every participant was welcomed and the experimenter handed over the study material and explained the task. Each session took approximately one hour.

Task. We asked the participants to group 69 DAGs with respect to their perceived similarity – multiple occurrences of a single DAG in different groups were allowed. Furthermore, we asked them to tag each group with the factors they used to build them. Finally, participants had to judge the easiness of forming the respective group (“How difficult or easy was it for you to create this group?”) and their confidence in the group’s consistency (“How doubtful or confident are you about the consistency of the DAGs in the group, i.e., would you create the same group again if you did this task again?”). The questions regarding easiness and confidence were to judge on a five-point Likert scale (“1 = very difficult/doubtful, 2 = difficult/doubtful, 3 = neutral, 4 = easy/confident, 5 = very easy/confident”). The formed groups provide the data needed to answer *RQ1* while the participants’ group tags provide the data to answer *RQ2*. For the task formulation we kept the one from Pandey et al. [33] since it captured exactly what we wanted to ask our participants. Moreover, the formulation was already pretested and successful in Pandey et al.’s study.

Card Sorting Methodology. Card sorting is a well-known methodology in psychology and human-computer interaction for externalizing mental models humans have about the environment they live in. Wood and Wood [46] define card sorting

as follows: *As the name implies, the method originally consisted of researchers writing labels representing concepts (either abstract or concrete) on cards, and then asking participants to sort (categorize) the cards into piles that were similar in some ways.* Humans group objects according to their perceived similarity into different categories. In this way, card sorting helps to uncover the structure of mental models. There are different methods to conduct card sorting. Researchers generally distinguish between open vs. closed sorting tasks and between paper-based and computer-supported card sorting [14]. In closed card sorting, participants have to sort the cards according to a given scheme, in open card sorting, the participants develop it themselves. The procedures for card sorting tasks sometimes differ considerably. Sometimes, the cards that have been assigned to a category are placed in a pile [46], so that participants do not shuffle them around on a canvas. Especially in computerized card sorting, it is often not possible to see all cards from which to choose at the same time [6, 33] which forces the study participants to compare the cards in memory. We used an open, paper-based card sorting since literature indicates that the paper-based approach yields more consistent results than the computerized one [14].

Study Setup and Materials. We used an empty meeting room with good lighting for conducting the study. The participant got the task sheet, the data sheet, sheets for building the groups, and sheets for tagging each group with the group building factors as well as for judging the easiness and the confidence. The task sheet contained the afore explained task. The data sheet consisted of the 69 randomly positioned DAGs. We decided to present our dataset on paper, so that the participants could see all data items at the same time. The order of the data items was kept the same for all participants to exclude order as a possible confounding variable. The participants had to write down the DAGs' IDs which belong to a group and give each group a unique identifier. Furthermore, they had to write down the tags as well as their easiness and confidence judgment together with the unique group identifier. The material is available on our website.

4 Analysis and Results

To analyze the collected data with respect to our research questions we used a mixed-approach involving a quantitative (RQ1, RQ2) and a qualitative (RQ2) analysis. The qualitative analysis provided the factors the participants tagged their formed groups with. The quantitative analysis resulted in the perceptual consensus over all participants as well as it served as a check of the participants' self-reported factors extracted in the qualitative analysis.

4.1 Quantitative Analysis (RQ1, RQ2)

We did a perceptual consensus calculation over all participants with complete data (16), i.e., participants who assigned each DAG to at least one group. The perceptual consensus of the perceived similarity served as a basis to find

out (1) whether the similarity perception of humans is consistent across individual people and (2) whether it is objectifiable with graph theoretical or visual properties.

To gain insights into the consistency and objectifiability of the similarity perception and to mitigate the potential bias – saying one thing and doing another – of self-reporting questions such as tagging we analyzed the perceived similarity consensus regarding which of the known graph theoretical and visual properties explain the clusters best. The mitigation potential resides in the perceived similarity consensus encapsulating what the participants really did.

The consensus easiness and confidence scores for each cluster provide information about the similarity consensus' perceived solidness and robustness. A high average easiness score means that the grouping is solid, thus, due to an easy assignment, it is less likely that a participant assigned a DAG randomly. The average confidence score reflects the participants' opinion whether they would form the same group again. A high score means that the grouping is robust since it is highly probable that it would look similar if the task were repeated.

Analysis. To build the perceptual consensus for the participants' similarity judgments, we calculated a pairwise perceptual distance between each pair of DAGs, based on the number of occurrences of each DAG pair in the same group and on the number of individual occurrences (for details cf. [33]). The perceptual distance calculation resulted in a 69×69 perceptual distance matrix (PDM). Like Pandey et al. [33] we did a hierarchical clustering, in our case with average linkage. We evaluated the correct number of clusters with the mean/median of number of groups and with the gap statistic [42]. The mean/median indicate the average number of participant-built groups and thus served as a reasonable estimator for the number of clusters. The gap statistic respects, like the individual groupings and similarity per se, the cluster similarity which made it to a further reasonable estimator. The hierarchical clustering result is the consensus grouping of all DAGs based on the similarity consensus contained in the PDM.

For the clusters' property analysis we determined various properties for each graph. Based on this we determined the dominating properties of the clusters as well as the cluster separating properties. Examples of the employed properties are: depth, visual symmetry, visual leaning, edge crossing – number and existence, edge length, number of nodes on a specific level, and the existence and the number of nodes having more than on parent node.

For the consensus of the easiness and confidence score we calculated an easiness and confidence value for each plot on the basis of the assumption that each plot inherits the easiness and confidence score of the participant-built groups it belongs to. Then we calculated an average easiness and confidence score for the hierarchical clustering result. For a detailed explanation please refer to [33].

Results. The gap statistic indicated that the data supports eight clusters. Both the mean and median of the number of built groups supported the indicated eight clusters (*mean* = 7.6, *median* = 8.0, *STD* = 2.6). As all three were similar, we decided to cut the tree into eight clusters. Figure 2 shows the resulting

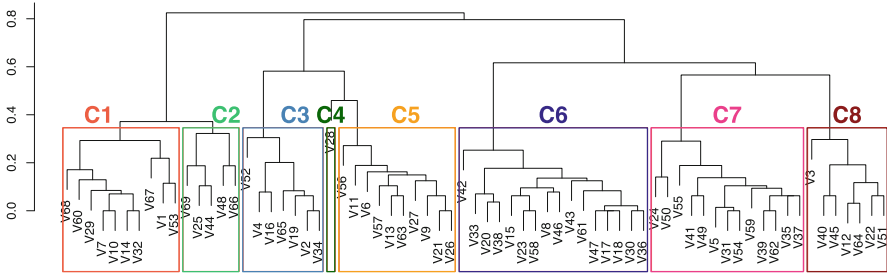


Fig. 2. Dendrogram resulting from hierarchical clustering with average linkage. The resulting eight clusters (C1–C8) are marked using colored boxes. (Color figure online)

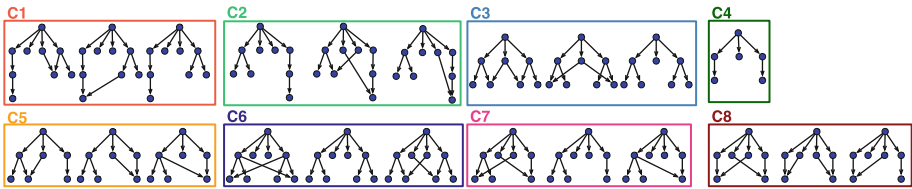


Fig. 3. Excerpts of the hierarchical clusters (cf. website complete clusters). (Color figure online)

dendrogram and the resulting clusters – marked using colored boxes. Excerpts of the hierarchical clusters are shown in Fig. 3. The entire clusters can be found on our website. The easiness and confidence scores of all hierarchical clusters are around 4.0 (cf. Table 1). This means that the participants on average found their groups *easy* to build and were *confident* they would look similar if they repeated the task. Consequently, this results in a good solidness and robustness of the consensus grouping.

The properties which distinguish the clusters best are the *depth* of the DAGs, the *number of nodes on a specific level* of the DAGs, and the *visual leaning* of the DAGs. Table 1 summarizes the properties of the clusters. Clusters C1 and C2 are identical in depth and number of nodes on each of their four levels. However, they are separated by the leaning. While the DAGs of C1 are left-skewed, those of C2 are right-skewed. The leaning separating the clusters C1 and C2 suggests that not the reflection of G6 (cf. Fig. 1) itself was apparent to the participants but rather a property which changed – the leaning (cf. Sect. 3.2).

Clusters C3, C4, and C5 have identical depth (3) as well as three nodes on the second level. The number of nodes on the third level separates these clusters. The depth separates the clusters C3, C4, C5 from C1, C2. Cluster C5 clearly shows that neither the reflection of G2 (cf. Fig. 1) itself nor a changed property mattered. It seems that the pure number of nodes dominates significantly over, e.g., node position (2 left, 1 right vs. reflected: 1 left, 2 right).

Clusters C6, C7, and C8 have identical depth (3) and four nodes on the second level. The number of nodes on the third level separates them. The number of nodes on the second level separates C6, C7, C8 from C3, C4, C5. C6, C7, C8 and C1, C2 are separated by depth. C7 shows that also the reflection of G4 itself (cf. Fig. 1) or a changed property, e.g., node position, did not matter.

Interestingly, *edges* and *edge crossings* – important factors of graph theory and graph aesthetics – seem not to matter to the participants. The excerpts of C3 and C5 in Fig. 3 clearly show: The edges had no influence on the similarity judgment of the participants. Otherwise DAGs with such different topology would not have been grouped together. The excerpt of C7 shows that the participants also did not really care about edge crossings. To conclude, we consider the consistency of the hierarchical clusters as high regarding graph theoretical and visual DAG properties. They are also well objectifiable with these properties.

Table 1. Properties of the DAGs in the clusters C1-C8 along with average easiness (Ease) and confidence (Conf) values for each cluster.

Cluster	Ease	Conf	DAG properties
C1	4.3	4.2	depth: 4 • number of nodes on level 2: 4; on level 3: 3; on level 4: 1 • leaning: left
C2	4.4	4.3	depth: 4 • number of nodes on level 2: 4; on level 3: 3; on level 4: 1 • leaning: right
C3	4.1	4.0	depth: 3 • number of nodes on level 2: 3; on level 3: 4
C4	4.1	4.1	depth: 3 • number of nodes on level 2: 3; on level 3: 2
C5	3.6	3.8	depth: 3 • number of nodes on level 2: 3; on level 3: 3
C6	3.7	3.7	depth: 3 • number of nodes on level 2: 4; on level 3: 4
C7	3.6	3.8	depth: 3 • number of nodes on level 2: 4; on level 3: 3
C8	3.8	3.9	depth: 3 • number of nodes on level 2: 4; on level 3: 2

4.2 Qualitative Analysis (RQ2)

We performed a thematic analysis of the participants' tags to reveal the factors they considered. We also analyzed the factors' importance based on the number of mentions of a specific factor. For this analysis we used the data of all 20 participants since it does not depend on whether a DAG was grouped or not.

Analysis. First, we transcribed the participants' tags by noting each tag together with how the participant used it, e.g., in a hierarchical manner. Additionally we collected the following data for the tags (henceforth called factors) of each participant: factor type (visual, graph theoretical), combined vs. single factors (e.g., number of levels vs. number of levels *and* number of nodes), number of considered factors, number of values per factor (e.g., number of edge crossings = 1, 2 and 3 → number of values = 3). We deemed the factor type as important since the graph theoretical properties are those which contain the information

relevant for comparison insights. However, we already know from graph readability research the significant influence of visual factors (e.g., edge crossing). Knowing those for visual comparison is beneficial for controlling their influence. We collected the other data as meta-information on the factors the participants used in order to learn more about the participants' usage of the factors.

Results. The individual transcriptions can be found on our website. Figure 4 shows the factors considered by the participants together with how often a factor was named. Multiple mentions of one and the same factor by one participant were not considered. In total, our participants used 27 distinct factors (cf. Fig. 4). Ten of these can be considered to be graph theoretical factors (yellow) and 15 to be visual factors (blue). Two of the used factors are neither graph theoretical nor visual (gray). Just five out of 20 participants used a combined factor and only two of these five used more than one combined factor. The most frequently combined factor was *number of nodes on a specific level* (five times); e.g., number of nodes on the second level = 3 and number of nodes on the third level = 4. Eight of the 27 factors were used by at least 20% of the participants (cf. Fig. 4, left). We will focus on these eight, for the other 20 factors please refer to Fig. 4, right.

The most important factors according to usage frequency were: *number of levels* (i.e., depth of the DAG), *number of nodes on a specific level*, *shape*, *arm/branch* (\equiv DAG sub-shape), *one parent node*, *edge crossing*, *child node(s) with > 1 parent node*, *visual leaning* (left: $\uparrow \wedge$ right: $\wedge \uparrow$). The factor *shape* is basically the convex hull of the DAG (\triangle). Regarding *shape* it is interesting to note that we could observe a coherence of *shape* with the *number of nodes on a specific level*. Participants, for instance, denoted a DAG such as $\wedge \wedge \wedge$ as “narrow/small pyramid” and a DAG such as $\wedge \wedge \wedge$ as “wide/large pyramid”. However, it is clear that this coherence is also influenced by the DAGs' layout. *Arm/branch* refers to the *shape of a DAG's sub-graph* ($\uparrow \wedge$). *Edge crossing* deals with crossings of the visualized edges ($\wedge \wedge$). The participants considered different types of edge crossings, e.g., presence of edge crossing or (un)resolvable edge crossings. The factors *one parent node* and *child node(s) with > 1 parent node* relate to the

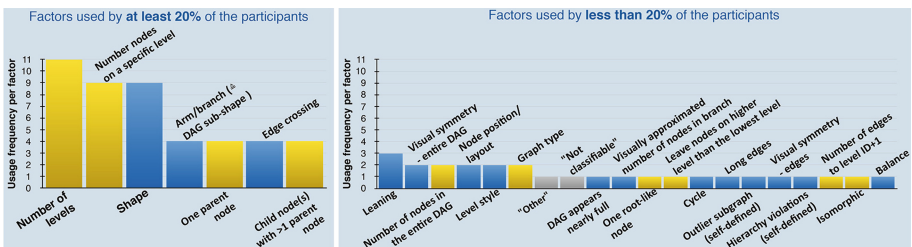







Fig. 4. Factors used by the participants (yellow: graph theoretical, blue: visual, gray: no type). Multiple mentions of the same factor by the same participant were excluded. (Color figure online)

number of nodes which are parent to another node (, ). Again, we could observe that participants used different types of these factors.

Interestingly, also the extracted factors substantiate that edges and edge crossings did not really matter (cf. Sect. 4.1). The factor *edge crossing* is one of the least used of the most important factors. Other edge related factors were used just once (cf. Fig. 4, right). Various individual groupings also support this, e.g.:    (factor: one parent left).

5 Discussion and Conclusion

We conducted a card sorting study to identify the factors influencing the similarity perception of DAGs to mitigate the present knowledge gap regarding this topic despite the vast presence of visual comparison tasks in various disciplines.

Both, the results of our quantitative and qualitative analysis point to *similar* factors which seem to dominantly influence similarity perception of DAGs, namely the *number of levels (depth)*, the *number of nodes on specific levels*, as well as *shape-related aspects* such as the visual leaning of a DAG. Herewith, we can be certain that the self-reported factors of the participants were not biased. The strong influence of shape is remarkable as in our case the spatial arrangement did not convey any additional information. This resulted in cases where structurally identical DAGs were assigned to different groups due to one being left-skewed or right-skewed. Being skewed to the left or right mainly played a role for the 4-level DAGs (cf. C1 and C2), most likely because it had a stronger influence on the overall shape as in the 3-level cases. Nevertheless, this observation supports previous results which found evidence that perception of graphs is sensitive to its spatial layout (cf. e.g., [18, 30]). Surprisingly, edge crossings – an important factor concerning the readability of graphs [36] – contrary to our expectations did not seem to have a strong impact on perceived DAG similarity. This is, for example, evident in the clusters C5 and C6 where no distinction between DAGs with and without edge crossings has been made (cf. Fig. 3). In the participants' statements we found soft evidence that they did not subconsciously resolve the edge crossing and therefore did not mention edge factors; on the contrary, the edges were not in the focus of the participants.

The fixed order of our data items did not lead to arbitrary groupings. The individual groupings and the consensus grouping are well objectifiable with DAG properties. We analyzed the individual groupings by checking the objectifiability of grouped consecutive data items (cf. website for details). The quantitative analysis shows the objectifiability of the consensus grouping (cf. Sect. 4.1).

In future work, it will be necessary to investigate how the identified factors and their importance varies across different graph sizes. It is, for instance, reasonable to assume that, for larger graphs, factors concerning details of a graph (e.g., number of parent nodes, number of nodes on a specific layer) decrease in importance while factors concerning the overall appearance (e.g., shape) increase. Regardless of that, our study provides first results which can contribute

to the design of comparative visualizations. Moreover, a better understanding of the factors which drive humans' similarity judgment may also be used towards developing perception-based graph similarity measures. Current notions of graph similarity such as graph isomorphism and edit distance (cf. [10]), descriptive statistics of graph structure measures such as degree distribution or diameter, or iterative approaches which assess the similarity of the neighborhood of nodes (e.g., [19, 21, 31]) rely purely on graph theoretical properties.

Besides understanding the individual factors we also deem it important to understand the strategies that participants employ while judging the similarity of data items. This will help to offer useful interactions with comparative visualizations. While our study was not specifically designed for this we could observe circumstantial evidence, as a byproduct from our transcription, that the participants used three distinctive strategies: Eleven participants chose a factor, grouped the entire dataset according to it, and then grouped the resulting groups into further sub-groups (*divide-and-conquer*). There were also seven participants who always *respected the entire dataset considering the factors one after the other*. Some of the participants chose all their factors in advance. Still others chose their factors in an ad hoc fashion; meaning, after having grouped the dataset according to a factor they thought about the next. Finally, there were two participants who did their grouping by considering just *one single factor*. More thorough investigations will be necessary to verify these observations.

To conclude, we consider the similarity perception of DAGs in visual comparison across people as consistent and well objectifiable with graph theoretical or visual properties. We find this substantiated by our quantitative and qualitative analysis. An in-depth analysis is subject to future research.

Acknowledgments. This work was financially supported by the Deutsche Forschungsgemeinschaft e.V. (DFG, LA 3001/2-1) and the Austrian Science Fund (FWF, I 2703-N31).

References

1. Archambault, D.: Structural differences between two graphs through hierarchies. In: Proceedings of Graphics, pp. 87–94. Canadian Information Processing Society (2009)
2. Archambault, D., Purchase, H.C., Pinaud, B.: Difference map readability for dynamic graphs. In: Brandes, U., Cornelsen, S. (eds.) GD 2010. LNCS, vol. 6502, pp. 50–61. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-18469-7_5
3. Bach, B., Pietriga, E., Fekete, J.D.: GraphDiaries: animated transitions and temporal navigation for dynamic networks. IEEE Trans. Vis. Comput. Graphics **20**(5), 740–754 (2014)
4. Beck, F., Burch, M., Diehl, S., Weiskopf, D.: The state of the art in visualizing dynamic graphs. In: Proceedings of EuroVis - STARs (2014)
5. Bremm, S., Von Landesberger, T., Heß, M., Schreck, T., Weil, P., Hamacher, K.: Interactive visual comparison of multiple trees. In: Proceedings of IEEE VAST, pp. 31–40 (2011)

6. Chaparro, B.S., Hinkle, V.D., Riley, S.K.: The usability of computerized card sorting: a comparison of three applications by researchers and end users. *J. Usability Stud.* **4**(1), 31–48 (2008)
7. Collins, C.M., Carpendale, S.: VisLink: revealing relationships amongst visualizations. *IEEE Trans. Vis. Comput. Graph.* **13**(6), 1192–1199 (2007)
8. Dwyer, T., Lee, B., Fisher, D., Quinn, K.I., Isenberg, P., Robertson, G., North, C.: A comparison of user-generated and automatic graph layouts. *IEEE Trans. Vis. Comput. Graph.* **15**(6), 961–968 (2009)
9. Fuchs, J., Isenberg, P., Bezerianos, A., Fischer, F., Bertini, E.: The influence of contour on similarity perception of star glyphs. *IEEE Trans. Vis. Comput. Graph.* **20**(12), 2251–2260 (2014)
10. Gao, X., Xiao, B., Tao, D., Li, X.: A survey of graph edit distance. *Pattern Anal. Appl.* **13**(1), 113–129 (2010)
11. Ghani, S., Elmqvist, N., Yi, J.S.: Perception of animated node-link diagrams for dynamic graphs. *Comput. Graph. Forum* **31**(3), 1205–1214 (2012)
12. Gleicher, M., Albers, D., Walker, R., Jusufi, I., Hansen, C.D., Roberts, J.C.: Visual comparison for information visualization. *Inf. Vis.* **10**(4), 289–309 (2011)
13. Goldstone, R.L., Son, J.Y.: Similarity. In: Holyoak, K.J., Morrison, R.G. (Eds.) *The Cambridge Handbook of Thinking and Reasoning* (2005)
14. Greve, G.: Different or alike? comparing computer-based and paper-based card sorting. *Int. J. Strateg. Innovative Mark.* **1**(1), 27–36 (2014)
15. Hadlak, S., Schumann, H., Schulz, H.J.: A survey of multi-faceted graph visualization. In: *Proceedings of EuroVis - STARs* (2015)
16. Holten, D., Van Wijk, J.J.: Visual comparison of hierarchically organized data. *Comput. Graph. Forum* **27**(3), 759–766 (2008)
17. Holten, D., van Wijk, J.J.: A user study on visualizing directed edges in graphs. In: *Proceedings of CHI*, pp. 2299–2308 (2009)
18. Huang, W., Hong, S.-H., Eades, P.: Layout effects on sociogram perception. In: Healy, P., Nikolov, N.S. (eds.) *GD 2005*. LNCS, vol. 3843, pp. 262–273. Springer, Heidelberg (2006). https://doi.org/10.1007/11618058_24
19. Jeh, G., Widom, J.: SimRank: a measure of structural-context similarity. In: *Proceedings of KDD*, pp. 538–543 (2002)
20. Kieffer, S., Dwyer, T., Marriott, K., Wybrow, M.: HOLA: Human-like orthogonal network layout. *IEEE Trans. Vis. Comput. Graph.* **22**(1), 349–358 (2016)
21. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. *J. ACM* **46**(5), 604–632 (1999)
22. Klippel, A., Hardisty, F., Weaver, C.: Star plots: how shape characteristics influence classification tasks. *Cartogr. Geogr. Inf. Sci.* **36**(2), 149–163 (2009)
23. Kobourov, S.G., Pupyrev, S., Saket, B.: Are crossings important for drawing large graphs? In: Duncan, C., Symvonis, A. (eds.) *GD 2014*. LNCS, vol. 8871, pp. 234–245. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45803-7_20
24. Körner, C.: Concepts and misconceptions in comprehension of hierarchical graphs. *Learn. Instr.* **15**(4), 281–296 (2005)
25. von Landesberger, T., Kuijper, A., Schreck, T., Kohlhammer, J., van Wijk, J., Fekete, J.D., Fellner, D.: Visual analysis of large graphs: State-of-the-art and future research challenges. *Comput. Graph. Forum* **30**(6), 1719–1749 (2011)
26. von Landesberger, T., Diel, S., Bremm, S., Fellner, D.W.: Visual analysis of contagion in networks. *Inf. Vis.* **14**(2), 93–110 (2015)
27. von Landesberger, T., Pohl, M., Wallner, G., Distler, M., Ballweg, K.: Investigating graph similarity perception: a preliminary study and methodological challenges. In: *Proceedings of VISIGRAPP*, pp. 241–250 (2017)

28. Lenz, O., Keul, F., Bremm, S., Hamacher, K., von Landesberger, T.: Visual analysis of patterns in multiple amino acid mutation graphs. In: Proceedings of IEEE VAST, pp. 93–102 (2014)
29. McGee, F., Dingliana, J.: An empirical study on the impact of edge bundling on user comprehension of graphs. In: Proceedings of AVI, pp. 620–627 (2012)
30. McGrath, C., Blythe, J., Krackhardt, D.: The effect of spatial arrangement on judgments and errors in interpreting graphs. *Soc. Netw.* **19**(3), 223–242 (1997)
31. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In: Proceedings of ICDE, pp. 117–128 (2002)
32. Novick, L.R.: The importance of both diagrammatic conventions and domain-specific knowledge for diagram literacy in science: the hierarchy as an illustrative case. In: Barker-Plummer, D., Cox, R., Swoboda, N. (eds.) *Diagrams 2006*. LNCS (LNAI), vol. 4045, pp. 1–11. Springer, Heidelberg (2006). https://doi.org/10.1007/11783183_1
33. Pandey, A.V., Krause, J., Felix, C., Boy, J., Bertini, E.: Towards understanding human similarity perception in the analysis of large sets of scatter plots. In: Proceedings of CHI, pp. 3659–3669 (2016)
34. Pekalska, E., Duin, R.P.W.: *The dissimilarity representation for pattern recognition: Foundations and applications* (2005)
35. Purchase, H.C., Pilcher, C., Plimmer, B.: Graph drawing aesthetics - created by users, not algorithms. *IEEE Trans. Vis. Comput. Graph.* **18**(1), 81–92 (2012)
36. Purchase, H.: Which aesthetic has the greatest effect on human understanding? In: DiBattista, G. (ed.) *GD 1997*. LNCS, vol. 1353, pp. 248–261. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63938-1_67
37. Purchase, H.C.: Metrics for graph drawing aesthetics. *Vis. Lang. Comput.* **13**(5), 501–516 (2002)
38. Purchase, H.C., Hoggan, E., Görg, C.: How important is the “Mental Map”? – An empirical investigation of a dynamic graph layout algorithm. In: Kaufmann, M., Wagner, D. (eds.) *GD 2006*. LNCS, vol. 4372, pp. 184–195. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70904-6_19
39. Purchase, H.C., McGill, M., Colpoys, L., Carrington, D.: Graph drawing aesthetics and the comprehension of UML class diagrams: An empirical study. In: Proceedings of Invis.au. pp. 129–137 (2001)
40. Tennekes, M., de Jonge, E.: Tree colors: color schemes for tree-structured data. *IEEE Trans. Vis. Comput. Graph.* **20**(12), 2072–2081 (2014)
41. Thornley, S., Marshall, R., Wells, S., Jackson, R.: Using directed acyclic graphs for investigating causal paths for cardiovascular disease. *J. Biometrics Biostatistics* **4**, 182 (2013)
42. Tibshirani, R., Walther, G., Hastie, T.: Estimating the number of clusters in a data set via the gap statistic. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **63**(2), 411–423 (2001)
43. Tominski, C., Forsell, C., Johansson, J.: Interaction support for visual comparison inspired by natural behavior. *IEEE Trans. Vis. Comput. Graph.* **18**(12), 2719–2728 (2012)
44. Vehlou, C., Beck, F., Weiskopf, D.: The state of the art in visualizing group structures in graphs. In: Proceedings of EuroVis - STARs (2015)
45. Welch, E., Kobourov, S.: Measuring symmetry in drawings of graphs. *Comput. Graph. Forum* **36**(3), 341–351 (2017)
46. Wood, J.R., Wood, L.E.: Card sorting: current practices and beyond. *J. Usability Stud.* **4**(1), 1–6 (2008)

GiViP: A Visual Profiler for Distributed Graph Processing Systems

Alessio Arleo^(✉), Walter Didimo, Giuseppe Liotta,
and Fabrizio Montecchiani

Università degli Studi di Perugia, Perugia, Italy
alessio.arleo@studenti.unipg.it, {walter.didimo,giuseppe.liotta,
fabrizio.montecchiani}@unipg.it

Abstract. Analyzing large-scale graphs provides valuable insights in different application scenarios. While many graph processing systems working on top of distributed infrastructures have been proposed to deal with big graphs, the tasks of profiling and debugging their massive computations remain time consuming and error-prone. This paper presents GiViP, a visual profiler for distributed graph processing systems based on a Pregel-like computation model. GiViP captures the huge amount of messages exchanged throughout a computation and provides an interactive user interface for the visual analysis of the collected data. We show how to take advantage of GiViP to detect anomalies related to the computation and to the infrastructure, such as slow computing units and anomalous message patterns.

1 Introduction

The analysis of large-scale graphs provides valuable insights in different application scenarios, including social networking, crime detection, content ranking and recommendations (see, e.g., [19, 37, 43, 60]). On the other hand, graph computations are often difficult to scale and parallelize, due to the inherent interdependencies within graph data. Furthermore, graph algorithms are usually iterative and hence poorly suited for popular Big Data processing systems such as Hadoop/MapReduce (see, e.g., [20, 42]). In response to these shortcomings, new frameworks based on the Think-Like-A-Vertex (TLAV) programming model have been proposed, such as Google's Pregel [43] and its open source counterpart Apache Giraph [19]. The idea behind the TLAV model is to provide a common vertex-centric programming interface, abstracting from low-level details of the distributed infrastructure. Graph processing systems based on the TLAV model outperform general purpose Big Data processing systems by improving locality and by demonstrating linear scalability [45]. In view of their effectiveness, these systems are being adopted by a growing number of applications. For example Apache Giraph is used in the contexts of social networking [37], fraud detection [60], and network visualization [8, 9].

We thank Maria Elisa Ganci for her contribution in the development of GiViP.

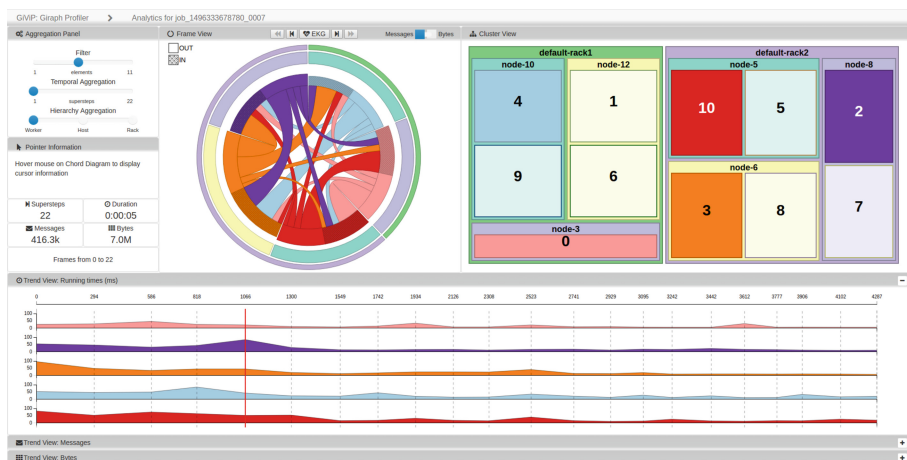


Fig. 1. The graphical interface of GiViP.

While many graph processing systems working on top of modern distributed infrastructures have been proposed to deal with large graphs, the tasks of profiling and debugging their massive computations remain time consuming and error-prone [31, 54]. Low-level profiling systems for distributed architectures exist [18, 39], but none of them is tailored to the needs of TLAV frameworks (or other types of distributed graph processing systems). For example, Hadoop Profiler [18] is designed to analyze CPU workloads of Apache Hadoop clusters [1], but it disregards the interaction between pairs of computing units, which is crucial in a TLAV framework. Indeed, algorithms written for TLAV-based graph processing systems usually rely on slim user-defined functions that do not require much CPU resources, but they may require huge numbers of messages and/or iterations to propagate the results of a local computation throughout the graph. A classical example is the TLAV implementation of the PageRank algorithm, which requires each vertex to iteratively execute a simple computation and to communicate the output to all its neighbors, until convergence is achieved [43]. Moreover, anomalies related to the distributed infrastructure may yield to unbalanced partitions of the input graph over the computing units which, in turn, leads to overloaded links in the distributed infrastructure. Similarly, a buggy implementation of an algorithm may yield to anomalous message patterns.

Contribution. In this paper we tackle the challenge of profiling massive computations that run on top of a TLAV-based graph processing system, and we provide a publicly available implementation of our approach¹, called GiViP, for Apache Giraph. Figure 1 shows a screenshot of the graphical interface of GiViP. The system collects the networked data related to messages exchanged by pairs of computing units throughout a specific computation, constructs suitable

¹ <http://givip.graphdrawing.cloud/>.

aggregations of these data, and presents to the user an interactive visual interface for exploring them. To demonstrate the effectiveness of our approach, we discuss key usage scenarios of GiViP in terms of resource profiling and detection of both computation- and infrastructure-related issues, such as overloaded computing units and anomalous message patterns. For reasons of space some material has been omitted and can be found in [7].

2 Background and Related Work

Background. The *Think-Like-A-Vertex (TLAV) programming model* provides a common vertex-centric programming interface, abstracting from low-level details of the computation and of the distributed infrastructure. Assuming (with no restrictions) that the input graph is directed, a *user-defined function* aims at updating the internal value of the vertex and/or of its outgoing edges. It takes as input data from the incoming edges of the vertex, while its output is communicated through the outgoing edges. Thus, each vertex exchanges *messages* only with its neighbors. Google’s Pregel [43] was the first published implementation of a TLAV framework. It is based on the Bulk-Synchronous Programming (BSP) model [62], which splits the computation into iterations called *supersteps*, with synchronization barriers occurring between consecutive supersteps. At each superstep the user-defined function is executed over the vertices of the graph, and the messages sent by a vertex during a superstep are received by its neighbors at the beginning of the next superstep. The computation halts after a number of rounds, or when a halting condition is met. Apache Giraph [19] is a popular Java-based TLAV framework built on Apache Hadoop [1] and originated as the open source counterpart of Pregel. Giraph exhibits additional features with respect to Pregel, but it is still based on the BSP model. A fundamental ingredient of large-scale graph processing systems is a preliminary partitioning operation that splits the input graph into parts assigned to different computing units. Good partitions often lead to improved performance, but expensive partitioning strategies may end up dominating the processing time. In Giraph, a basic computing unit is called *worker*, and each computer, or *host*, can run multiple workers. In large clusters, hosts are grouped into *racks*. Giraph provides a default hash-based partitioning algorithm to assign each vertex of the input graph to a worker. Different strategies can be employed by overriding suitable methods of the library. We point to the survey by McCune et al. [45] for further references and explanations about TLAV frameworks. In particular, Apache Hama [56] and GPS [55] are Pregel-like systems, hence our approach can be adapted for them.

Debuggers, profilers, and monitoring tools. While modern distributed platforms transparently handle the hassles related to the distributed infrastructure, debugging and profiling computations, as well as monitoring and optimizing the underneath infrastructure, remain challenging tasks. Hadoop Profiler [18] is a tool to analyze CPU workloads for Apache Hadoop clusters. The statsd-jvm-profiler [39] enables the analysis of memory usage, garbage collection, and the aggregate execution time of each function within Apache Hadoop clusters.

Both these tools work at low-level, without distinguishing between concurrent computations running on the same cluster. We also mention high performance computing (HPC) profilers such as Gprof [30] and VTune [52], which sample the execution of a computation and analyze the time spent on each part of the code. BigDebug [31] is a tool offering interactive, real-time debugging primitives for computations running on Apache Spark [2, 66], an in-memory engine for Apache Hadoop. Graft [54] offers a graphical interface to debug TLAV programs, and it is implemented for Apache Giraph. None of these tools offers resource profiling features. CloudGazer [59] is a visualization system that allows users to monitor cloud-based networks. This system has provided valuable inspiration for our work but its focus is different from ours, as it is directed towards the optimization of cloud-based infrastructures in order to reduce energy consumption and to increase the quality of service.

Time series visualizations. Profiling a computation involves the analysis of time-varying parameters. Classic charts for time-series data include line charts [48], small multiples [61], stacked graphs [17], horizon graphs [53], and braided graphs [38] (see also [33]). Javed et al. [38] compared these types of visualizations in a user study with local and global tasks on samples with up to 8 simultaneous time series. They observed that shared-space visualizations excel at comparisons with a local visual span, while split-space techniques are more robust against high numbers of concurrent time series for tasks that need large visual spans. More compact iconic representations can also be used when dealing with many simultaneous time series, at the expenses of a less intuitive temporal encoding; see, e.g., the survey by Ward [64] and the user study by Fuchs et al. [28]. Also, several application-driven systems have been proposed that make use of ad-hoc visualizations. Examples are: ThermalPlot [58], for the visualization of multi-attribute time-series data highlighting significant developments over time; CloudLines [40], for time-based representations of large and dynamic event data sets; LiveRAC [46], for the visualization of large collections of system management time-series data with hundreds of parameters; ThemeRiver [32], for visualizing thematic variations over time within a large collection of documents; LifeLines [47], for representing personal histories.

Dynamic graph drawing. In GiViP the communication among workers is conveniently modeled as a graph whose edges are weighted based on the amount or the size of messages exchanged between pairs of workers during a superstep. Hence, our problem intersects the rich literature on dynamic graph drawing (see, e.g., [11, 16, 21, 27]). Nonetheless, the topology of our communication graph is unlikely to change over time, as each worker communicates with workers that manage the neighbors of its vertices, regardless of the superstep.

3 The GiViP System

3.1 Tasks and Requirements

The tasks that guided the design of GiViP are conceived having in mind the analysis of the resources used by computations running on top of Pregel-like graph

processing systems; thus, they substantially differ from the common objectives of low-level distributed profilers. The main tasks are as follows.

T1 *Analyze the performance trend of a computation in terms of running time and traffic load.* This task is relevant to evaluate the scalability of a distributed algorithm and to detect possible bottlenecks. High running times may be alleviated by scaling up the resources of the cluster; at the same time, adding computational units may even increase the traffic load (as it increases the input fragmentation). Also, peaks of resources may be caused by software or hardware faults, and a deeper inspection of the data may shed more light on the problem.

T2 *Analyze the traffic between pairs of computing units (workers, hosts, racks).* This is useful to detect overloaded links at different levels of the cluster hierarchy, and to estimate the quality of the graph partitioning algorithm. Note that links between racks are usually slower than links between hosts in the same rack, which are in turn slower than links between workers in the same host.

T3 *Analyze data aggregated at different computing scale and time scale.* Aggregating data at different computing scales is needed because the size of a cluster can vary from a few hosts in the same rack, up to many hosts within multiple racks. By aggregating data at different time scales we mean the possibility of aggregating sequences of supersteps. This is particularly useful for executions that span hundreds or thousands of supersteps. The number of supersteps taken by an execution usually depends on several variables such as the structural properties of the input graph, the type of algorithm, and the halting condition.

We also considered two requirements aimed at simplifying the usage of the system: **R1** *Avoid user code instrumentation.* While distributed debuggers often require specific instructions to be incorporated in the user code (see, e.g., [54]), this is commonly avoided in distributed profilers. This feature facilitates the portability of the code in production environments, as the profiler can be switched off without recompiling the user code. **R2** *Allow remote access to the user interface.* This is essential when the user has no direct access to the computing platform (e.g., when using PaaS products such as Amazon EC2), but instead uses a remote connection or a Web interface to access the cluster.

3.2 Data Model and Data Aggregation

We now describe how data are organized in GiViP and how they can be aggregated to support scalability in the visual interface.

Data model. The inclusion relationships between workers, hosts, and racks (see Sect. 2) are represented by an inclusion tree T , which does not change over time. A Giraph computation, called *job*, is spread over a sequence of $k > 0$ synchronized supersteps. For each superstep i (for $i = 1, \dots, k$), starting at instant s_i , the data collected by GiViP are modeled as a weighted directed graph (digraph) $G_i = (V_i, E_i)$. Each vertex v of G_i represents a worker and has a weight $t_i(v)$, denoting the time taken by the worker to complete its task in superstep i . The synchronization barriers between supersteps imply that

$s_i + \max_{v \in V_i} \{t_i(v)\} \leq s_{i+1}$. Also, each directed edge (u, v) has two weights, $m_i(uv)$ and $b_i(uv)$, denoting the number of messages and their total size (in bytes) sent from u to v during superstep i , respectively.

Data aggregation. GiViP allows two types of data aggregation. *Temporal aggregation* consists of grouping consecutive supersteps in a single *frame*. Let s_i and s_j ($i \leq j$) be the first and the last superstep of a frame f_{ij} . The system computes a weighted digraph $G_{ij} = (V_i \cup V_{i+1} \cup \dots \cup V_j, E_i \cup E_{i+1} \cup \dots \cup E_j)$. For example, if a computation takes 10,000 supersteps, a temporal aggregation with 100 supersteps per frame results in a sequence of 100 digraphs. The weight of each vertex v of G_{ij} is $t_{ij}(v) = \sum_{z=i}^j t_z(v)$, and for each edge (u, v) of G_{ij} , we have $m_{ij}(uv) = \sum_{z=i}^j m_z(uv)$ and $b_{ij}(uv) = \sum_{z=i}^j b_z(uv)$.

Hierarchy aggregation merges workers based on their membership in the same host or rack. Aggregating data in a hierarchical fashion is a well established method to alleviate visual clutter and to support scalability [26]. Consider a weighted digraph G_{ij} (possibly with $i = j$). A hierarchy aggregation at the *host level* computes a weighted digraph G_{ij}^H as follows. For each host $h \in T$ we have a vertex v in G_{ij}^H , whose weight $t_{ij}^H(v)$ equals the sum of the weights of all vertices of G_{ij} that belongs to h . Similarly, we have an edge (u, v) in G_{ij}^H if there is at least an edge in G_{ij} between a vertex in the host of u and a vertex in the host of v . The weights $m_{ij}^H(uv)$ and $b_{ij}^H(uv)$ are computed as the sum of the corresponding weights over all edges between a vertex in the host of u and a vertex in the host of v . Analogously, a hierarchy aggregation at the *rack level* computes a graph G_{ij}^R by aggregating workers in the same rack. A hierarchy aggregation at the *worker level* trivially corresponds to $G_{ij}^W = G_{ij}$.

In what follows, for a weighted digraph G_{ij} we assume that $i \leq j$. If $i = j$, then no temporal aggregation has been performed. To simplify the notation, we may omit the superscript (W, H, or R) that specifies the hierarchy aggregation level, if this is not relevant for the discussion and does not create ambiguities.

3.3 Visualization Paradigm and Interface

The interface of GiViP allows users to interactively explore the networked data associated with a computation. The interface is divided into four main views, which we call *Aggregation Panel*, *Cluster View*, *Trend View*, and *Frame View* (see Fig. 1). The Trend View and the Frame View mainly support tasks **T1** and **T2**, respectively. The Aggregation Panel supports task **T3**. The Cluster View conveys the hierarchical structure of the computing cluster and is used to filter elements of this hierarchy. The three views are coordinated and highly interactive. Each worker is associated with a unique color, which is consistently used in all views. We used color schemes offered by the D3.js library [13].

Aggregation Panel. It contains controls that have impact on both the Trend View and the Frame View. A temporal aggregation can be performed by using a slider to set the size of each frame. A hierarchy aggregation can be set by means of a three-state switch. In addition, the user can filter the computing units based

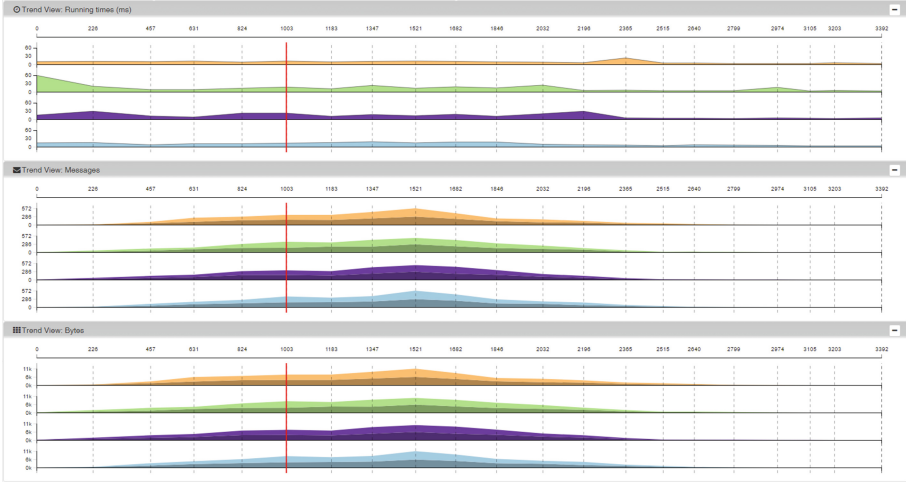


Fig. 2. The Trend View.

on the total amount of messages they exchange, so to hide those units that have a smaller impact in terms of traffic load. Finally, this panel contains some high-level statistics such as the total running time and the number of supersteps taken by the computation, and the total number of exchanged messages and bytes.

Cluster View. Interacting with this view allows focusing only on a subset of computational units, by filtering out workers, hosts, and racks. Filtered workers disappear from both the Frame and the Trend View. If a host (rack) is filtered out, then all its workers (hosts) are filtered out. The inclusion tree T is shown by means of a squarified treemap [15]. By clicking on a tile, the corresponding computational unit is filtered in or out based on its current state. The size of a tile is proportional to the number of vertices of the input graph assigned to the corresponding computational unit. This is helpful in two ways. First, the user can decide to filter those units that contain fewer vertices. Second, the user has an immediate feeling of whether the graph partitioning algorithm produced a balanced partition or not. Recall that Giraph’s default partitioning algorithm guarantees balanced partitions, but different strategies can be employed to optimize other criteria, such as minimizing inter-worker links [63].

Trend View. For each computing unit, this view shows the evolution throughout the computation of running time, number of exchanged messages, and amount of exchanged bytes. (A computing unit is a worker, a host, or a rack, depending on the hierarchy aggregation level.) We encode this information as a set of three small multiples [61], vertically stacked and with a shared time axis, see Fig. 2. We recall that Javed et al. [38] experimentally observed that split-space visualizations are particularly robust against various concurrent time series for tasks that need large visual spans, which is exactly our setting (T1). The first small multiple shows the running time over all the computation frames.

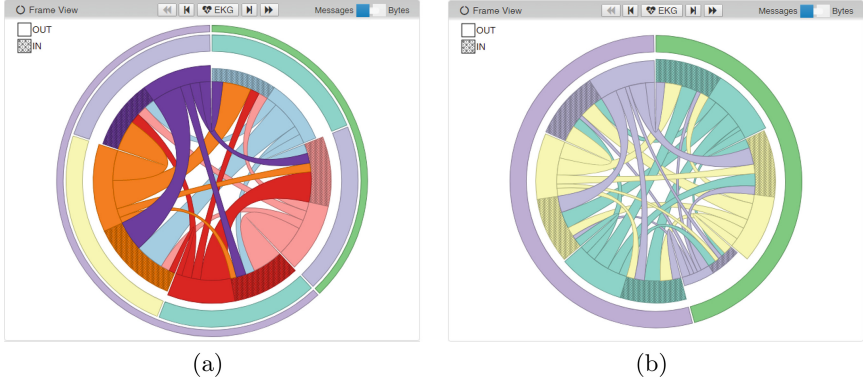


Fig. 3. The Frame View, hierarchy aggregation at (a) worker and (b) host level.

Each single chart is an area chart showing the evolution for the corresponding computing unit. The second small multiple shows the number of messages exchanged over all computation frames. Each single chart is a stacked area chart (also known as stream graph) that shows both the incoming and the outgoing messages of the corresponding computing unit, and thus which also conveys the total number of messages. The incoming messages are depicted with a regular texture to darken the original color assigned to the computing unit. Distinguishing between incoming and outgoing messages is useful because each worker is responsible only for the outgoing edges incident to its vertices, while the incoming edges play a role in the amount of messages that will be received in the next frame. The third small multiples is similar to the previous one but the traffic load is measured in terms of bytes. Each of the three small multiples is enclosed in a collapsible panel. Finally, the shared time axis is paginated and initialized responsively with a number of frames per page to guarantee an adequate resolution. As a rule of thumb, a display with 1920×1080 px allows up to 50 frames, while 20 frames guarantees a pleasant distribution of the labels.

Frame View. Here we depict the traffic load between pairs of computing units. Let $G_{i,j}$ be the digraph corresponding to a frame $f_{i,j}$. As discussed in Sect. 2, the topology of $G_{i,j}$ does not depend on the specific frame $f_{i,j}$. Indeed, as we observed in our experiments, $G_{i,j}$ is usually a complete graph, especially if the hierarchy aggregation is set to the host or rack level. On the other hand, the edge weights may significantly differ depending on the frame. These observations motivate a network visualization method that privileges the user mental map preservation [5, 51], and that is conceived to effectively encode edge weights. We implemented an enhanced version of the chord diagram available in [13], as shown in Fig. 3. A chord diagram is a circular layout in which the vertices of the graph are arranged as thick circle arcs, and the edges are shown with ribbons connecting pairs of arcs. The size of a ribbon encodes the quantitative information associated with the corresponding edge, and thus each circle arc

is long enough to accommodate the ends of its ribbons. Chord diagrams are effectively adopted in various contexts such as comparative genomics [41], urban mobility trajectories [29], and others [3]. Also, they can be extended to support hierarchical data sets (see, e.g., [6, 36]), as in our case. We use concentric circles to encode the hierarchy levels. Circle arcs representing workers (hosts) in the same host (rack) appear consecutively around the circle. If the hierarchy aggregation is set at the worker level, then the three levels of the hierarchy are simultaneously shown; see Fig. 3(a). If the data are aggregated at host or rack level, then only two levels or one level are shown, respectively; see Fig. 3(b). The main novelties introduced by our enhanced chord diagram are: (i) the use of heuristics for crossing minimization inspired by the literature on circular layouts (see, e.g., [10, 22, 24, 57]), and (ii) a bimodal orientation of the edges in which the incoming and the outgoing edges of each vertex form two contiguous intervals [23].

Edge crossing minimization. Edge crossings are a form of visual clutter that deteriorates the readability of a drawing [49, 50, 65]. We use a variant of the heuristic by Baur and Brandes [10] to minimize edge crossings (the optimization problem is NP-complete [44]); it deals with the constraints imposed by the inclusion tree T and with (dynamic) edge weights. Our algorithm takes as input the graph G_{1k} , where k is the number of supersteps of the computation, and computes a unique circular order of the vertices, used for the visualization of all graphs G_{ij} . This is crucial for the user mental map preservation, especially when the visualization changes due to filtering or aggregations.

Bimodal orientation. In the chord diagram, the orientation of an edge is encoded by coloring its ribbon with the same color as the source vertex. In addition, we split the circle arc of a vertex into two intervals, one for the incoming edges and one for the outgoing. The length of each interval reflects the total weight of the corresponding edges, which facilitates the comparison between incoming and outgoing traffic at a computing unit. To avoid crossings between adjacent edges, the outgoing edges of a vertex always follow the incoming edges in clockwise order. The interval for the incoming edges is filled with a regular pattern to darken its original color (as in the Trend View). Furthermore, in our chord diagram, a self-loop is encoded by thickening its vertex (circle arc) proportionally to its weight; this helps to understand the amount of traffic within the same unit.

Interaction. Every aggregation or filtering operation is immediately reflected in all views. Changes in the Trend and Frame Views are smoothed by animated transitions, which help in preserving the user mental map. The time axis of the Trend View is anchored with a slider to browse the frames of the computation. When the user releases the cursor of the slider, the chord diagram smoothly changes the width of its ribbons, so to highlight significant changes. By mouse hovering on the various visualizations, details are immediately shown through pop-ups. For example, by hovering a ribbon of the chord diagram, the number of messages (and bytes) associated with the edge is displayed, or by hovering an area chart, the corresponding value of the diagram is shown.

3.4 Architecture and Implementation Notes

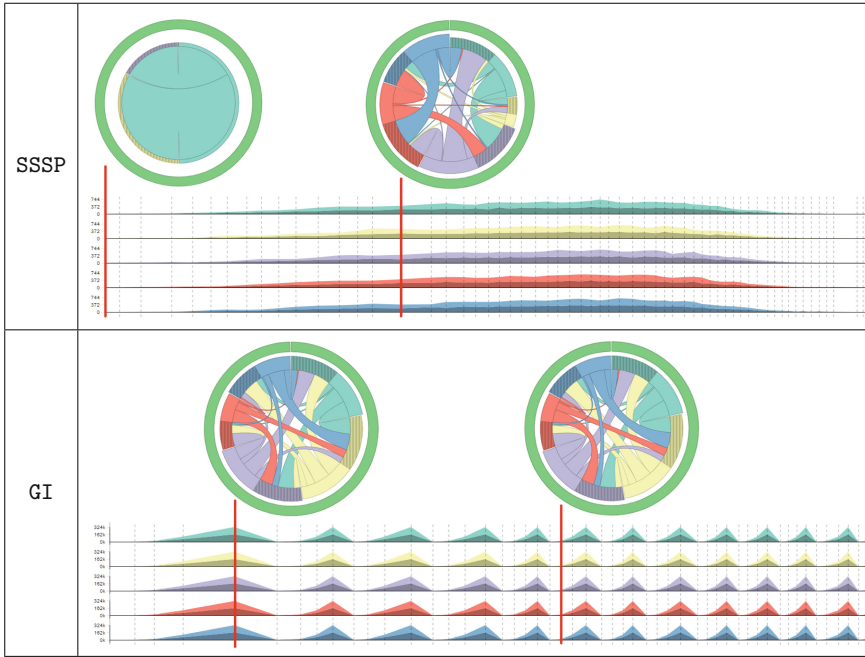
The architecture of GiViP is composed of two main modules. The MESSAGE SNIFFER collects all data that need to be analyzed. It is realized as a patch for Giraph’s source code and can be deployed without user code instrumentation (**R1**). The data are collected asynchronously so to minimize the impact of this module on the computation. Some experiments (on 20 computations) showed that using our patch does not slow down a Giraph job by more than 36%, and only by 7.5% on average. As a comparison, other systems to monitor parallel and distributed algorithms have an overhead around 5% [4, 14]. Although GiViP is not meant to be used in production environments, these numbers suggest the profiling activity does not seriously affect the running time of a computation. The VISUAL ANALYZER has a Java back-end that aggregates and stores the collected data in a MySQL database, and that provides a RESTful API to access the data. The front-end runs in a Web browser (**R2**) and implements the GUI of GiViP. It is coded in HTML/CSS/Javascript and exploits the D3.js [13] library.

4 Usage Scenarios

We discuss the effectiveness of GiViP in key scenarios covering all tasks of Sect. 3.1. We used two clusters, depending on the experiment. One is an Amazon EC2 cluster with 1 rack, 10 hosts, and 20 workers. The other is a cluster of commodity machines at our university with 1 rack, 6 hosts, and 11 workers.

Scenario 1: Resource profiling. Distributed algorithms are characterized by the trend of the performance parameters throughout a computation. This trend can be regarded as the “heartbeat” of the algorithm, as it is only partially affected by the input graph and by the cluster configuration. Deviations from the expected behavior should raise a warning on possible hardware or software failures. We performed experiments that show how GiViP effectively conveys the heartbeats of some algorithms. This feature can be used both for a visual confirmation of a successful execution and for didactic purposes. We considered four algorithms: **Single-Source Shortest-Path (SSSP)** and **Page Rank (PR)** [43] are well-known graph algorithms, available in the set of examples provided by the Apache Giraph library; **GILA (GI)** [8] and **MultiGILA (MGI)** [9] are TLAV implementations of a force-directed algorithm and of a multilevel force-directed algorithm, respectively. We ran these algorithms on two graphs, **cti** and **Gnutella31**. The first is a mesh with 16,840 vertices and 48,232 edges, while the second is a peer-to-peer communication network with 62,686 vertices and 147,892 edges. Table 1 refers to graph **cti**. It shows the small multiples representing the exchanged messages (with a hierarchy aggregation at the host level, and after filtering out some hosts with lower traffic), and two representative snapshots of the chord diagram. The traffic load of SSSP follows a Gaussian-like trend, since the algorithm is based on a flooding technique that reaches its peak when all vertices know their shortest distance from the source vertex. From the first chord diagram, one can see that there is only one host that generates

Table 1. Resource profiling for SSSP and GI on graph *cti*.



messages in the first superstep, which means that this host contains the source vertex. The messages of GI follows a periodic pattern, where each period represents a controlled flooding in which the coordinates of a vertex u are broadcast to all vertices within a fixed topological distance from u . The chord diagrams at different supersteps look very similar, which tells that the percentages of traffic exchanged between pairs of hosts are stable, even if the total number of messages changes. The traffic load of PR is flat, as the algorithm is based on a set of identical supersteps in which each vertex updates its internal status and communicates with all its neighbors. The chord diagram does not change among different supersteps, as a further witness of this constant behavior. Algorithm MGI alternates computation phases with a periodic trend and phases with flat trend, as a consequence of the multilevel scheme. For example the initial supersteps (concerned with the coarsening phase of algorithm) are very short and generate few messages; the corresponding chord diagrams highlight unbalanced links, due to the fact that only some vertices of the graph are activated in this phase of the algorithm.

Scenario 2: Anomalous message patterns. A deviation from the expected heartbeat of an algorithm should warn the user of a possible issue in the computation. To see this, we injected a bug in the SSSP algorithm and we ran a new experiment. According to the algorithm, if during a superstep there is a vertex

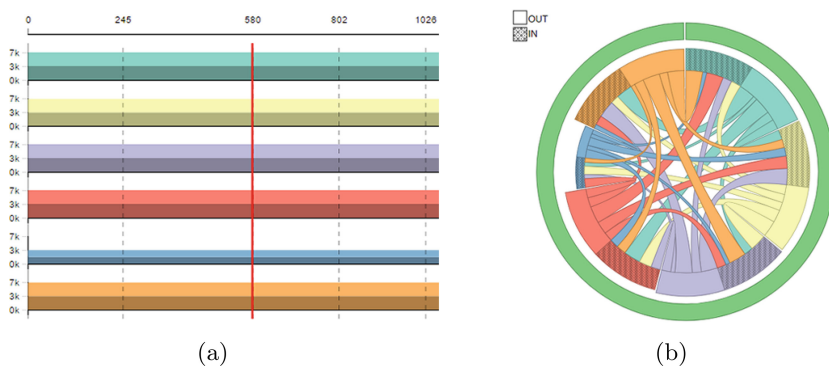


Fig. 4. Scenario 2: Anomalous message pattern for algorithm SSSP. Detail of the (a) Trend View and of the (b) Frame View with hierarchy aggregation at the host level.

u that decreases its best-known distance from the source vertex, then u sends a message to all its neighbors. We added a piece of code that delivers messages to the neighbors of u also if its best-known distance does not change. This causes unnecessary messages, but does not affect the correctness of the algorithm; thus, such a bug would not be discovered by just looking at the output of the computation. From the Trend View, the user immediately observes a flat trend of messages, which deviates from the expected Gaussian-like heartbeat (see Fig. 4). The chord diagram shows that there are no overloaded links, i.e., the anomalous messages are distributed among the hosts. This confirms that the problem comes from an implementation bug, rather than from a hardware issue.

Scenario 3: Slow computing units. Due to the synchronization barriers between supersteps, if a computing unit is significantly slower than the others, it causes a bottleneck for the entire computation. Since the resource management is transparent to the user, such an event is difficult to spot by using default tools such as the Hadoop dashboard and the Giraph counters. In contrast, a slow computing unit can be easily detected in our Trend View. Also, since the problem is usually due to a faulty or overloaded host, an aggregation at the host level may expose the problem.

We ran the PR algorithm on the `4e1t` graph (a mesh with 15,607 vertices and 45,878 edges). We used our local cluster, whose hosts run within a virtualized environment. We limited the percentage of usable CPU for one of them (while keeping the virtualized hardware the same for all the hosts). The Trend View clearly shows the existence of a host whose running time is way higher than the others (indeed, the others are barely visible). Also, the Frame View shows that the slow host (red) handles an amount of messages similar to that of the others. Hence, the poor performance cannot be accounted to a difference in the traffic load, but should be searched in the host conditions (see Fig. 5).

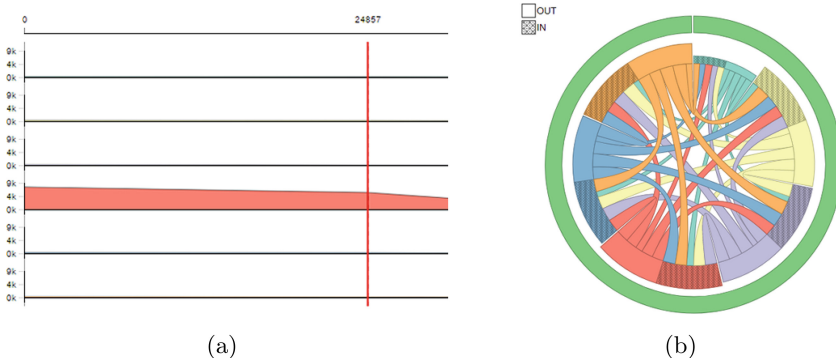


Fig. 5. Scenario 3: Slow host for algorithm PR. Detail of the (a) Trend View and of the (b) Frame View with hierarchy aggregation at the host level. (Color figure online)

5 Discussion and Future Work

We presented GiViP, the first visual profiler for distributed algorithms on Pregel-like graph processing systems, and showed that it can be used in several situations to detect different computation- and infrastructure-related issues. One limitation of GiViP is concerned with the Frame View, that requires the usage of filters and/or aggregations if more than a few tens of vertices need to be displayed. This is due to fact that the chord diagram suffers from edge clutter. Although it is uncommon to have more than a few tens of computers allocated for a single computation, one can think of investigating alternative graph visualizations, such as matrix-based ones (see, e.g., [12, 25, 34]), to improve scalability in our application domain. We also plan to extend GiViP with the possibility of executing temporal queries [35], and of aggregating sequences of supersteps by computation phase. In addition, it would be interesting to collect events from the cluster’s resource manager, to detect possible failures of the resource containers.

References

1. <http://hadoop.apache.org/>. Accessed 10 June 2017
2. <https://spark.apache.org/>. Accessed 10 June 2017
3. <http://www.circos.ca>. Accessed 10 June 2017
4. Hpc toolkit (2011). <http://hpc toolkit.org/index.html> Accessed 22 Aug 2017
5. Archambault, D., Purchase, H.C., Pinaud, B.: Animation, small multiples, and the effect of mental map preservation in dynamic graphs. *IEEE Trans. Vis. Comput. Graph.* **17**(4), 539–552 (2011)
6. Argyriou, E.N., Symvonis, A., Vassiliou, V.: A fraud detection visualization system utilizing radial drawings and heat-maps. In: Laramée, R.S., Kerren, A., Braz, J. (eds.) *IVAPP 2014*, pp. 153–160. SciTePress (2014)
7. Arleo, A., Didimo, W., Liotta, G., Montecchiani, F.: GiViP: a visual profiler for distributed graph processing systems. *ArXiv e-prints* <http://arxiv.org/abs/1708.07985> (2017)

8. Arleo, A., Didimo, W., Liotta, G., Montecchiani, F.: A distributed multilevel force-directed algorithm. In: Hu, Y., Nöllenburg, M. (eds.) GD 2016. LNCS, vol. 9801, pp. 3–17. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_1
9. Arleo, A., Didimo, W., Liotta, G., Montecchiani, F.: Large graph visualizations using a distributed computing platform. *Inf. Sci.* **381**, 124–141 (2017)
10. Baur, M., Brandes, U.: Crossing reduction in circular layouts. In: Hromkovič, J., Nagl, M., Westfechtel, B. (eds.) WG 2004. LNCS, vol. 3353, pp. 332–343. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30559-0_28
11. Beck, F., Burch, M., Diehl, S., Weiskopf, D.: A taxonomy and survey of dynamic graph visualization. *Comput. Graph. Forum* **36**(1), 133–159 (2017)
12. Behrisch, M., Bach, B., Hund, M., Delz, M., von Rügen, L., Fekete, J., Schreck, T.: Magnostics: image-based search of interesting matrix views for guided network exploration. *IEEE Trans. Vis. Comput. Graph.* **23**(1), 31–40 (2017)
13. Bostock, M., Ogievetsky, V., Heer, J.: D³ data-driven documents. *IEEE Trans. Vis. Comput. Graph.* **17**(12), 2301–2309 (2011)
14. Braun, B., Qin, H.: ddtrace: rich performance monitoring in distributed systems
15. Bruls, M., Huizing, K., van Wijk, J.J.: Squarified treemaps. In: de Leeuw, W.C., van Liere, R. (eds.) IEEE TCVG 2000. pp. 33–42. Eurographics Association (2000)
16. Burch, M., Vehlou, C., Beck, F., Diehl, S., Weiskopf, D.: Parallel edge splatting for scalable dynamic graph visualization. *IEEE Trans. Vis. Comput. Graph.* **17**(12), 2344–2353 (2011)
17. Byron, L., Wattenberg, M.: Stacked graphs - geometry & aesthetics. *IEEE Trans. Vis. Comput. Graph.* **14**(6), 1245–1252 (2008)
18. CERN: Hadoop profiler (2016). <https://github.com/cerndb/Hadoop-Profiler>. Accessed 10 June 2017
19. Ching, A., Edunov, S., Kabiljo, M., Logothetis, D., Muthukrishnan, S.: One trillion edges: graph processing at Facebook-scale. *PVLDB* **8**(12), 1804–1815 (2015)
20. Cohen, J.: Graph twiddling in a mapreduce world. *Comput. Sci. Eng.* **11**(4), 29–41 (2009)
21. Crnovrsanin, T., Chu, J., Ma, K.: An incremental layout method for visualizing online dynamic graphs. *J. Graph Algorithms Appl.* **21**(1), 55–80 (2017)
22. Dehkordi, H.R., Eades, P., Hong, S., Nguyen, Q.H.: Circular right-angle crossing drawings in linear time. *Theor. Comput. Sci.* **639**, 26–41 (2016)
23. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, Englewood Cliffs (1999)
24. Dogrusoz, U., Belviranlı, M.E., Dilek, A.: CiSE: a circular spring embedder layout algorithm. *IEEE Trans. Vis. Comput. Graph.* **19**(6), 953–966 (2013)
25. Elmqvist, N., Do, T.N., Goodell, H., Henry, N., Fekete, J.D.: ZAME: Interactive large-scale graph visualization. In: *IEEE PacificVis 2008*, pp. 215–222 (2008)
26. Elmqvist, N., Fekete, J.D.: Hierarchical aggregation for information visualization: overview, techniques, and design guidelines. *IEEE Trans. Vis. Comput. Graph.* **16**(3), 439–454 (2010)
27. Frishman, Y., Tal, A.: Online dynamic graph drawing. *IEEE Trans. Vis. Comput. Graph.* **14**(4), 727–740 (2008)
28. Fuchs, J., Fischer, F., Mansmann, F., Bertini, E., Isenberg, P.: Evaluation of alternative glyph designs for time series data in a small multiple setting. In: Mackay, W.E., Brewster, S.A., Bødker, S. (eds.) 2013 ACM SIGCHI, pp. 3237–3246. ACM (2013)
29. Gabrielli, L., Rinzivillo, S., Ronzano, F., Villatoro, D.: From Tweets to semantic trajectories: mining anomalous urban mobility patterns. In: Nin, J., Villatoro, D. (eds.) *CitiSens 2013*. LNCS (LNAI), vol. 8313, pp. 26–35. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04178-0_3

30. Graham, S.L., Kessler, P.B., McKusick, M.K.: Gprof: a call graph execution profiler. *ACM SIGPLAN Not.* **39**(4), 49–57 (2004)
31. Gulzar, M.A., Interlandi, M., Yoo, S., Tetali, S.D., Condie, T., Millstein, T.D., Kim, M.: BigDebug: debugging primitives for interactive big data processing in spark. In: *ICSE 2016*, pp. 784–795. ACM (2016)
32. Havre, S., Hetzler, B., Nowell, L.: Themeriver: visualizing theme changes over time. In: *IEEE InfoVis 2000*, pp. 115–123. IEEE (2000)
33. Heer, J., Bostock, M., Ogievetsky, V.: A tour through the visualization zoo. *Commun. ACM* **53**(6), 59–67 (2010)
34. Henry, N., Fekete, J., McGuffin, M.J.: NodeTrix: a hybrid visualization of social networks. *IEEE Trans. Vis. Comput. Graph.* **13**(6), 1302–1309 (2007)
35. Hochheiser, H., Shneiderman, B.: Dynamic query tools for time series data sets: timebox widgets for interactive exploration. *Inform. Vis.* **3**(1), 1–18 (2004)
36. Holten, D.: Hierarchical edge bundles: visualization of adjacency relations in hierarchical data. *IEEE Trans. Vis. Comput. Graph.* **12**(5), 741–748 (2006)
37. Jackson, J.: Facebook’s graph search puts Apache Giraph on the map (2013). <http://www.pcworld.com/article/2046680/facebooks-graph-search-puts-apache-giraph-on-the-map.html/>. Accessed 10 June 2017
38. Javed, W., McDonnel, B., Elmqvist, N.: Graphical perception of multiple time series. *IEEE Trans. Vis. Comput. Graph.* **16**(6), 927–934 (2010)
39. Johnson, A.: Introducing statsd-jvm-profiler: a JVM profiler for hadoop (2015). <https://github.com/cerndb/Hadoop-Profiler>. Accessed 10 June 2017
40. Krstajic, M., Bertini, E., Keim, D.: CloudLines: compact display of event episodes in multiple time-series. *IEEE Trans. Vis. Comput. Graph.* **17**(12), 2432–2439 (2011)
41. Krzywinski, M., Schein, J., Birol, I., Connors, J., Gascoyne, R., Horsman, D., Jones, S.J., Marra, M.A.: Circos: an information aesthetic for comparative genomics. *Genome Res.* **19**(9), 1639–1645 (2009)
42. Lumsdaine, A., Gregor, D.P., Hendrickson, B., Berry, J.W.: Challenges in parallel graph processing. *Parallel Process. Lett.* **17**(1), 5–20 (2007)
43. Malewicz, G., Austern, M.H., Bik, A.J.C., Dehnert, J.C., Horn, I., Leiser, N., Czajkowski, G.: Pregel: a system for large-scale graph processing. In: *ACM SIGMOD 2010*, pp. 135–146. ACM (2010)
44. Masuda, S., Kashiwabara, T., Nakajima, K., Fujisawa, T.: On the NP-completeness of a computer network layout problem. In: *IEEE International Symposium on Circuits and Systems*, pp. 292–295 (1987)
45. McCune, R.R., Weninger, T., Madey, G.: Thinking like a vertex: a survey of vertex-centric frameworks for large-scale distributed graph processing. *ACM Comput. Surv.* **48**(2), 25:1–25:39 (2015)
46. McLachlan, P., Munzner, T., Koutsofios, E., North, S.: LiveRAC: interactive visual exploration of system management time-series data. In: *2008 ACM SIGCHI*, pp. 1483–1492. ACM (2008)
47. Plaisant, C., Milash, B., Rose, A., Widoff, S., Shneiderman, B.: LifeLines: visualizing personal histories. In: *1996 ACM SIGCHI*, pp. 221–227. ACM (1996)
48. Playfair, W.: *The Commercial and Political Atlas: Representing, by Means of Stained Copper-plate Charts, the Progress of the Commerce, Revenues, Expenditure and Debts of England During the Whole of the Eighteenth Century*. Printed by T. Burton for J. Wallis, etc; 3rd edn. (1801)
49. Purchase, H.C.: Effective information visualisation: a study of graph drawing aesthetics and algorithms. *Interact. Comput.* **13**(2), 147–162 (2000)

50. Purchase, H.C., Carrington, D.A., Alder, J.A.: Empirical evaluation of aesthetics-based graph layout. *Empirical Softw. Eng.* **7**(3), 233–255 (2002)
51. Purchase, H.C., Hoggan, E., Görg, C.: How important is the “Mental Map”? – an empirical investigation of a dynamic graph layout algorithm. In: Kaufmann, M., Wagner, D. (eds.) *GD 2006*. LNCS, vol. 4372, pp. 184–195. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70904-6_19
52. Reinders, J.: *VTune Performance Analyzer Essentials*. Intel Press (2005)
53. Saito, T., Miyamura, H.N., Yamamoto, M., Saito, H., Hoshiya, Y., Kaseda, T.: Two-tone pseudo coloring: Compact visualization for one-dimensional data. In: *2005 IEEE InfoVis*, pp. 173–180. IEEE (2005)
54. Salihoglu, S., Shin, J., Khanna, V., Truong, B.Q., Widom, J.: Graft: a debugging tool for Apache Giraph. In: *ACM SIGMOD 2015*, pp. 1403–1408. ACM (2015)
55. Salihoglu, S., Widom, J.: GPS: a graph processing system. In: *SSDBM 2013*, pp. 22:1–22:12. ACM (2013)
56. Seo, S., Yoon, E.J., Kim, J., Jin, S., Kim, J., Maeng, S.: HAMA: an efficient matrix computation with the mapreduce framework. In: *CloudCom 2010*, pp. 721–726. IEEE (2010)
57. Six, J.M., Tollis, I.G.: A framework for circular drawings of networks. In: Kratochvíl, J. (ed.) *GD 1999*. LNCS, vol. 1731, pp. 107–116. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-46648-7_11
58. Stitz, H., Gratzl, S., Aigner, W., Streit, M.: ThermalPlot: visualizing multi-attribute time-series data using a thermal metaphor. *IEEE Trans. Vis. Comput. Graph.* **22**(12), 2594–2607 (2016)
59. Stitz, H., Gratzl, S., Krieger, M., Streit, M.: CloudGazer: a divide-and-conquer approach to monitoring and optimizing cloud-based networks. In: *IEEE PacificVis 2015*, pp. 175–182. IEEE (2015)
60. Tang, J.: Graph mining with Apache Giraph (2013). https://www.slideshare.net/Hadoop_Summit/tang-june26-205pmroom210cv2, Accessed 10 June 2017
61. Tufte, E.: *The Visual Display of Quantitative Information*. Encyclopedia of Mathematics and its Applications. Graphics Press, Cheshire (1983)
62. Valiant, L.G.: A bridging model for parallel computation. *Commun. ACM* **33**(8), 103–111 (1990)
63. Vaquero, L.M., Cuadrado, F., Logothetis, D., Martella, C.: Adaptive partitioning for large-scale dynamic graphs. In: *IEEE ICDCS 2014*, pp. 144–153. IEEE (2014)
64. Ward, M.O.: *Multivariate data glyphs: principles and practice*. Handbook of Data Visualization. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-33037-0_8
65. Ware, C., Purchase, H.C., Colpoys, L., McGill, M.: Cognitive measurements of graph aesthetics. *Inform. Vis.* **1**(2), 103–110 (2002)
66. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M.J., Shenker, S., Stoica, I.: Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: *NSDI 2012*, p. 2. USENIX Association (2012)

Drawing Big Graphs Using Spectral Sparsification

Peter Eades, Quan Nguyen^(✉), and Seok-Hee Hong

The School of Information Technologies, University of Sydney, Sydney, Australia
{peter.eades, quan.nguyen, seokhee.hong}@sydney.edu.au

Abstract. Spectral sparsification is a general technique developed by Spielman *et al.* to reduce the number of edges in a graph while retaining its structural properties. We investigate the use of spectral sparsification to produce good visual representations of big graphs. We evaluate spectral sparsification approaches on real-world and synthetic graphs. We show that spectral sparsifiers are more effective than random edge sampling. Our results lead to guidelines for using spectral sparsification in big graph visualization.

1 Introduction

The problem of drawing very large graphs is challenging and has motivated a large body of research (see [15] for a survey). As the number of vertices and edges becomes larger, layout algorithms become less effective. Further, runtime is increased both at the layout stage and at the rendering stage. Recent work (for example [20]) approaches the problem by replacing the original graph with a “proxy graph”. The proxy graph is typically much smaller than the original graph, and thus layout and rendering is easier. The challenge for the proxy graph approach is to ensure that the proxy graph is a good representation of the original graph; for visualization, we want the drawing of the proxy graph to be *faithful* [21] to the original graph.

In this paper we examine a specific proxy graph approach using *spectral sparsification* as introduced by Spielman *et al.* [1]: roughly speaking, the *spectrum* (that is, the eigenvalues of the Laplacian; see [10]) of the proxy graph approximates the spectrum of the original graph. Since the spectrum is closely related to graph-theoretic properties that are significant for graph drawing, this kind of proxy seems to promise faithful drawings.

We report results of an empirically investigation of the application of spectral sparsification to graph drawing. Specifically, we consider two closely related spectral sparsification techniques, one deterministic and one stochastic. We consider the quality of drawings so produced, using real-world and synthetic data sets. Quality is evaluated using the shape-based proxy graph metrics [20]. The results of spectral sparsification are compared to sparsifications obtained by simple random edge sampling. Our investigation confirms the promise of spectral sparsification, and shows that (overall) it is better than simple random edge sampling.

Section 2 recalls the proxy graph approach, and shape-based quality metrics for large graph drawing. Section 3 describes the application of spectral sparsification to graph visualization. Section 4 presents our experiments with spectral sparsification. The results of these experiments are presented and discussed in Sect. 5. Section 6 concludes.

2 Background

Proxy Graphs and Sparsification. The proxy graph approach is described in Fig. 1: for a given input graph G , a proxy graph G' and a drawing D' of G' are computed. The proxy graph represents G but is simpler and/or smaller than G in some sense. The user sees the drawing D' of G' , and does not see a drawing of the original graph G . However, if G' is a “good” representation of G , then D' is an adequate visualization of G in that the user can see all the structure of G in the drawing D' .

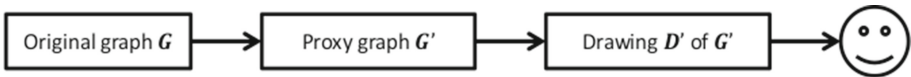


Fig. 1. In the *proxy graph approach*, the user sees a “proxy” of the original graph.

If G' is a subgraph of G , and the edge density of G' is smaller than the edge density of G , then we say that G' is a *sparsification* of G . Sparsification is the most common kind of proxy.

Sparsification has been extensively investigated in Graph Mining [9, 16, 19] (see survey [14]). Typically, sparsification is achieved by some kind of stochastic sampling. The most basic sparsification method is *random edge sampling (RE)*: each edge is chosen independently with probability p [23]. This and many other simple stochastic strategies have been empirically investigated in the context of visualization of large graphs [20, 29]. In this paper we apply a more sophisticated graph sparsification approach to visualization: the *spectral* sparsification work of Spielman *et al.* [1, 25, 26].

Shape-Based Quality Metrics. Traditional graph drawing metrics such as *edge bends*, *edge crossings*, and *angular resolution* are based on the *readability* of graphs; these metrics are good for small scale visualisation but become meaningless beyond a few hundred nodes [22]. For large graphs, *faithfulness metrics* are more important: informally, a drawing D of a graph G is *faithful* insofar as D determines G , that is, insofar as the mapping $G \rightarrow D$ is invertible.

Here we use *shape-based faithfulness metrics* [7]. The aim of these metrics is to measure how well the “shape” of the drawing represents the graph. For large graphs, such as in Fig. 2, the shape of the drawing is more significant than the number of edge bends and edge crossings. To make this notion more precise, we

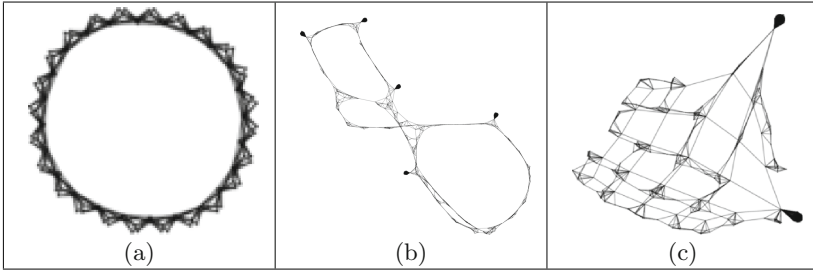


Fig. 2. The graphs *can_144*, *cN1031M22638*, and *gN733M62509*, drawn using FM³. Note that each has a distinctive shape.

use “shape graphs”. Given a set P of n points, a *shape graph* $S(P)$ is a graph with vertex set P such that the edge of $S(P)$ define the “shape” of P in some sense. Examples of shape graphs are the *Euclidean minimum spanning tree (EMST)*, the *relative neighbourhood graph (RNG)*, and the *Gabriel graph (GG)* [27].

Suppose that $G = (V, E)$ is a graph and P is a set of points in the plane, and each vertex $u \in V$ is associated with a point $p(u) \in P$. Denote the set of neighbours of u in G by $N_G(u)$, and the set of neighbours of $p(u)$ in the shape graph $S(P)$ by $N_{S(P)}(p(u))$. We say that

$$Jaccard(S(P), G) = \frac{1}{|V|} \sum_{u \in V} \frac{|N_G(u) \cap N_{S(P)}(p(u))|}{|N_G(u) \cup N_{S(P)}(p(u))|}$$

is the *Jaccard similarity* between the shape graph $S(P)$ and G . If D is a drawing of G then the (*shape-based*) *quality* of D is $Q(D, G) = Jaccard(P, G)$, where P is the set of vertex locations in the drawing D . Similarly, if D' is a drawing of a sparsification G' of G , then the (*shape-based*) (*proxy*) *quality* of D' is $Q(D', G) = Jaccard(P', G)$, where P' is the set of vertex locations in the drawing D' . Note that if u does not occur in D' , we consider that $N'(u) = \emptyset$. For more details, see [8].

3 The Spectral Sparsification Approach to Large Graph Drawing

First we describe some of the terminology and concepts of spectral graph theory. More details are in standard texts; for example, [5, 10]¹. The *adjacency matrix* of an n -vertex graph $G = (V, E)$ is the $n \times n$ matrix A , indexed by V , such that $A_{uv} = 1$ if $(u, v) \in E$ and $A_{uv} = 0$ otherwise. The *degree matrix* D of G is the diagonal matrix with where D_{uu} is the degree of vertex u . The *Laplacian* of G is $L = D - A$. The *spectrum* of G is the list $\lambda_1, \lambda_2, \dots, \lambda_n$ of eigenvalues of L . It can be shown that L has real nonnegative eigenvalues [5], and we assume that $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$; straightforward computation shows that $\lambda_1 = 0$.

¹ Beware: much of the terminology in spectral graph theory is not standardised.

The spectrum of a graph is closely related to many structural properties of the graph:

Connectivity: The number of connected components of G is largest value of i for which $\lambda_i = 0$ [5]. Roughly speaking, a larger value of λ_2 indicates a more highly connected graph and is related to the diameter of the graph. If $\lambda_2 > 0$ then it is called the *algebraic connectivity* of G [11].

Clusters: *Spectral clustering* involves the projection of the graph using its smallest eigenvalues. Spectral clustering solves a relaxation of the *ratio cut* problem, that is, the problem of dividing a graph into clusters to minimise the ratio between the number of inter-cluster edges and the cluster size [28]. Informally, the ratio cut problem seeks to find clusters of similar size so that the coupling between clusters is minimised.

Stress: The spectrum solves a kind of constrained stress problem for the graph. More specifically, the Courant-Fischer theorem (see [5]) implies that

$$\lambda_i = \min_{x \in X_i} \sum_{(u,v) \in E} (x_u - x_v)^2, \tag{1}$$

where X_i is the set of unit vectors orthogonal to the first $(i - 1)$ eigenvectors. The minimum is achieved when x is an eigenvalue corresponding to λ_i . Note that the right hand side of Eq. (1) is a kind of stress function.

Commute distance: The average time that a random walker takes to travel from vertex u to vertex v and return is the *commute distance* between u and v . Eigenvalues are related to random walks in the graph, and thus to commute distances (see [17]).

Spielman and Teng [26], following Benczur and Karger [3], first introduced the concept of “spectral approximation”. Suppose that G is an n -vertex graph with Laplacian L , and G' is an n -vertex subgraph of G with Laplacian L' . If there is an $\epsilon > 0$ such that for every $x \in R^n$,

$$(1 - \epsilon) \frac{x^T L' x}{x^T x} \leq \frac{x^T L x}{x^T x} \leq (1 + \epsilon) \frac{x^T L' x}{x^T x}, \tag{2}$$

then G' is an ϵ -spectral approximation of G . Using the Courant-Fischer Theorem [5] with (2), one can show that if G' is an ϵ -spectral approximation of G then the eigenvalues and eigenvectors of G' are close to those of G . The importance of this is that spectral approximation preserves the structural properties listed above.

Spielman and Teng first showed that every n -vertex graph has a spectral approximation with $O(n \log n)$ edges [26]. The following theorem is one such result:

Theorem 1 ([26]). *Suppose that G is an n -vertex graph and $\frac{1}{\sqrt{n}} \leq \epsilon \leq 1$. Then with probability at least $\frac{1}{2}$, there is an ϵ -spectral approximation G' of G with $O(\frac{1}{\epsilon} n \log n)$ edges.*

Further research of Spielman *et al.* refines and improves spectral sparsification methods (see [1]). These results have potential for resolving scale issues in graph visualisation by reducing the size of the graph while retaining its (spectral) structure. However, the *practical impact* of these results for graph visualization is not clear, because of large constants involved.

The proof of Theorem 1 is essentially a stochastic sampling method, using the concept of “effective resistance”. Suppose that we regard a graph $G = (V, E)$ as an electrical network where each edge is a $1 - \Omega$ resistor, and a current is applied. The voltage drop over an edge (u, v) is the *effective resistance* r_{uv} of (u, v) . Effective resistance in a graph is closely related to commute distance, and can be computed simply from the Moore-Penrose inverse [2] of the Laplacian. If L^\dagger is the Moore-Penrose inverse of L and $(u, v) \in E$, then $r_{uv} = L_{uu}^\dagger + L_{vv}^\dagger - 2L_{uv}^\dagger$.

We next describe two graph drawing algorithms, both variants of algorithms of Spielman *et al.* [1]. Each takes a graph G and an integer m' , and computes a sparsification G' with m' edges, then draws G' .

SSS (Stochastic Spectral Sparsification) randomly selects edges with probability proportional to their resistance value. Let E' be the edge set from m' random selections. Let G' be the subgraph of G induced by E' ; draw G' .

DSS (Deterministic Spectral Sparsification). Let E' consist of the m' of largest effective resistance. Let G' be the subgraph of G induced by E' ; draw G' .

In both DSS and SSS, the sparsified graph can be drawn with any large-graph layout algorithm.

4 The Experiments

The driving hypothesis for this paper is that for large graphs, spectral sparsification gives good proxy graphs for visualization. To be more precise, we define the *relative density* of the sparsification G' for a graph G to be $\frac{m'}{m}$, where G has m edges and G' has m' edges. Note that a proxy with higher relative density should be a better approximation to the original graph; thus we expect that drawings of the proxy with higher relative density should have better quality.

Since spectral sparsification (approximately) preserves the eigenvalues, we hypothesize that both SSS and DSS are better than RE. Further, we expect that the difference becomes smaller when the relative density is larger. To state this precisely, let D'_{SSS} (respectively D'_{RE}) denote the drawing obtained by SSS (respectively RE). We say that $\frac{Q(D'_{\text{SSS}}, G)}{Q(D'_{\text{RE}}, G)}$ is the *quality ratio* of SSS; similarly define the quality ratio of DSS. We expect that the quality ratio of both SSS and DSS is greater than 1. Further, we expect that the quality ratio for both algorithms tends to 1 as relative density tends to 1.

We implemented DSS, SSS and RE in Java, on top of the OpenIMAJ toolkit [13]. In particular, we used OpenIMAJ to compute the Moore-Penrose inverse. The experiments were performed on a Dell XPS 13 laptop, with an i7 Processor, 16 GB memory and 512 GB SSD. The laptop was running Ubuntu 16.04 with 20 GB swap memory. The computation of the Moore-Penrose inverse

used Java 8, with a specified 16 GB heap. We used multiple threads to speed up the resistance computation.

We used three data sets. The first set of graphs is taken from “defacto-benchmark” graphs, including the Hachul library, Walshaw’s Graph Partitioning Archive, the sparse matrices collection [6] and the network repository [24]. These include two types of graphs that have been extensively studied in graph drawing research: grid-like graphs and scale-free graphs. The second set is the GION data set [18]; this consists of RNA sequence graphs that are used for the analysis of repetitive sequences in sequencing data; these graphs have been used in previous experiments. They are locally dense and globally sparse, and generally have distinctive shapes. The third set consists of randomly generated graphs that contain interesting structures that are difficult to model with sparsification. Specifically, we generated a number of “black-hole graphs”, each of which consists of one or more large and dense parts (so-called “black holes”), and these parts connect with the rest of the graph by relatively few edges. These relatively few edges *outside* the “black holes” determine the structure of the graph. Such graphs are difficult to sparsify because sampling strategies tend to take edges from the dense “black holes” and miss the structurally important edges. Tuesday, December 19, 2017 at 9:06 am Figures 2(b) and (c) are black-hole graphs. Details of the graphs that we used are in Table 1.

Table 1. Data sets

graph	$ V $	$ E $	type
can_144	144	576	grid
G_15	1785	20459	scaleg
G_2	4970	7400	grid
G_3	2851	15093	grid
G_4	2075	4769	scaleg
mm_0	3296	6432	grid
nasa1824	1824	18692	grid
facebook01	4039	88234	scaleg
offlights	2939	15677	scaleg
soc_h	2426	16630	scaleg
yeastppi	2361	7182	scaleg

(a) Benchmark graphs

graph	$ V $	$ E $
graph_1	5452	118404
graph_2	1159	6424
graph_3	7885	427406
graph_4	5953	186279
graph_5	1748	13957
graph_6	1785	20459
graph_7	3010	41757
graph_8	4924	52502

(b) GION graphs

graph	$ V $	$ E $
cN377M4790	377	4790
cN823M14995	823	14995
cN1031M226386	1031	22638
gN285M2009	285	2009
gN733M62509	733	62509
gN1080M17636	1080	17636
gN4784M38135	4784	38135

(c) Black-hole graphs

We sparsify these input graphs to a range of relative density values: from small (1%, 2%, 3%, 4%, 5%, 10%) to medium and large (15%, 20%, \dots , 100%), using SSS, DSS, and RE.

For layout, we use the FM^3 algorithm [12], as implemented in OGDF [4]. However, we also confirmed our results using FM^3 variants (see Sect. 5.1).

We measured quality of the resulting visualizations by proxy quality metrics $Q(D, G)$ described in Sect. 2. For shape graphs, we used GG , RNG , and $EMST$;

the results for these three shape graphs are very similar, and here we report the results for *GG*.

5 Results from the Experiments

First we describe typical examples of the results of our experiments, using the graphs illustrated in Fig. 2; these are a relatively small defacto-benchmark graph *can_144*, and two black-hole graphs *cN1031M22638* and *gN733M62509*.

Sparsifications of *cN1031M22638* using RE, DSS, and SSS at relative densities of 3% and 15% are in Fig. 3. At relative density of 3%, both RE and SSS give poor results; the drawings do not show the structure of the graph. However, DSS gives a good representation. At relative density 15%, both DSS and SSS are good, while RE remains poor. A similar example, with relative densities of 1% and 10% for the black-hole graph *gN733M62509*, is in Fig. 4.

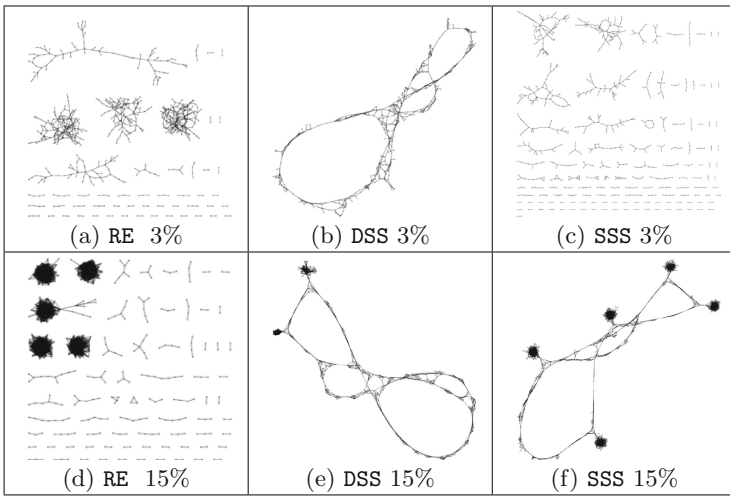


Fig. 3. Sparsifications of the graph *cN1031M22638* at relative densities 3% and 15%.

While the results for *cN1031M22638* and *gN733M62509* are typical, some results did not fit this mold. For *can_144*, see Fig. 5; here RE and SSS give poor representations, even at very high relative density (40%). However, all three algorithms give good representations at relative density 50%.

5.1 Quality: Results and Observations

Figure 6 shows the quality metrics for the three data sets for all three algorithms. The *x*-axis shows relative densities from 1% to 95%; the *y*-axis shows quality measures of the proxies.

We make the following five observations from the results.

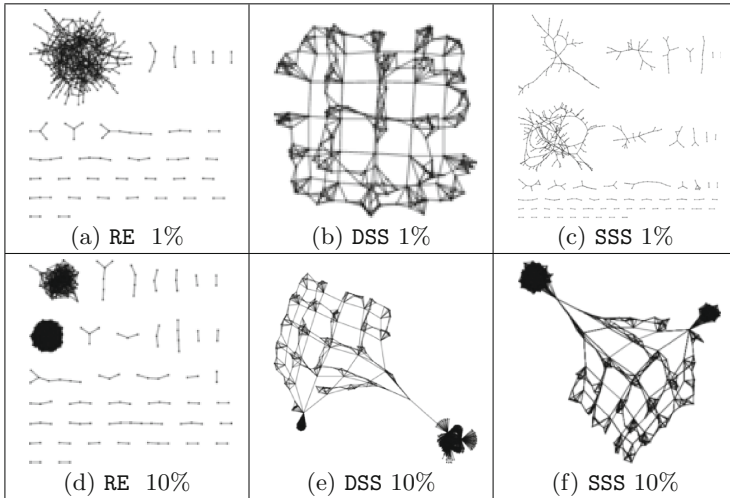


Fig. 4. Sparsifications of the graph $g_{N733M62509}$ at relative densities 1% and 10%.

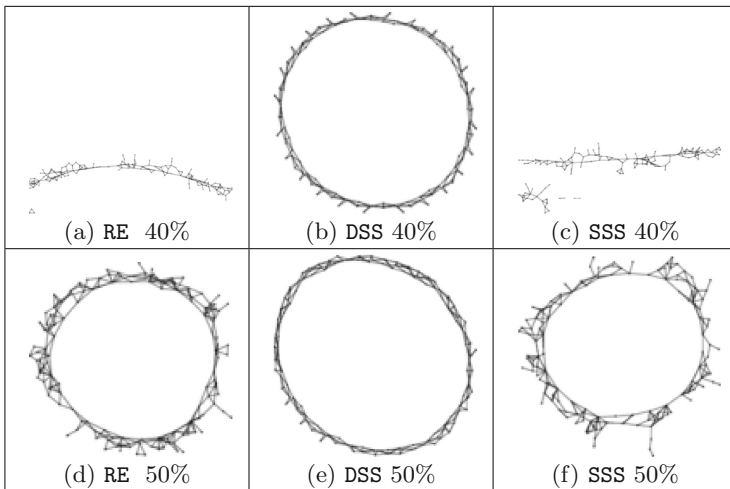


Fig. 5. Sparsifications of the graph can_{144} at relative densities 40% and 50%.

1. **Quality increases with relative density.** In general, quality increases as relative density increases. For many graphs there is a more interesting pattern: quality mostly increases up to a limit, achieved at a relative density between 10% and 30%, and then stays steady. Some of the defacto-benchmark graphs do not show this pattern: they show close to linear improvement in quality with density all the way up to 95%.
2. **Spectral sparsification is better than random edge sampling.** Figure 7 depicts the quality ratio (y -axis) for DSS and SSS for each data set, over

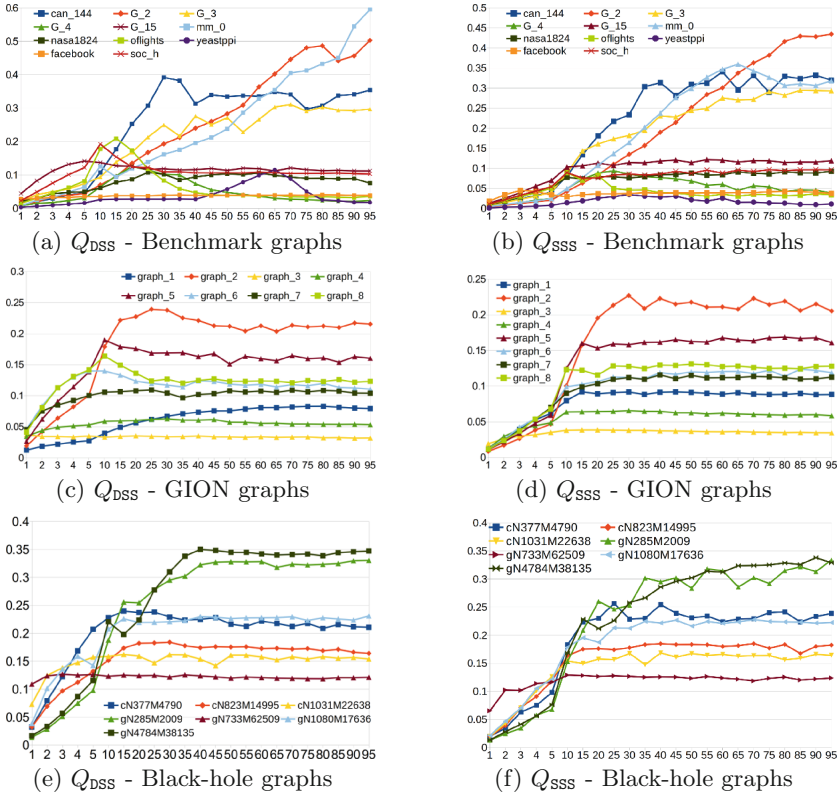


Fig. 6. Proxy quality metrics of output of DSS, SSS and RE: (a) and (b) defacto-benchmark graphs, (c) and (d) GION graphs, (e) and (f) black-hole graphs. The metrics use FM³ layout and GG shape graphs. (Color figure online)

relative density from 1% to 95%. Note that the quality ratio is significant in most cases, especially at low relative density. For example, DSS metrics are around 200 times better than RE, and sometimes much more (for the yeast dataset it is about 400).

For most of the graphs, the quality ratio decreases as the relative density increases. Quality ratio is best for relative density smaller than 10%. When the relative density is more than 15%, RE may be slightly better than DSS for a few graphs, such as defacto-benchmark graphs *mm_0* graph (light blue), and *G_2* (red). Interestingly, *soc_h* and *oflights* show a peak at around 10% and 15% before a drop for larger relative density.

- Sparsification is better for grid-like graphs than for scale-free graphs.** Figure 8(a) shows the quality change for DSS, SSS, and RE with density, over the grid-like and scale-free defacto-benchmark graphs. Note that average values for DSS and SSS are better than the average value for RE when the relative density is less than 35%. When relative density is greater than

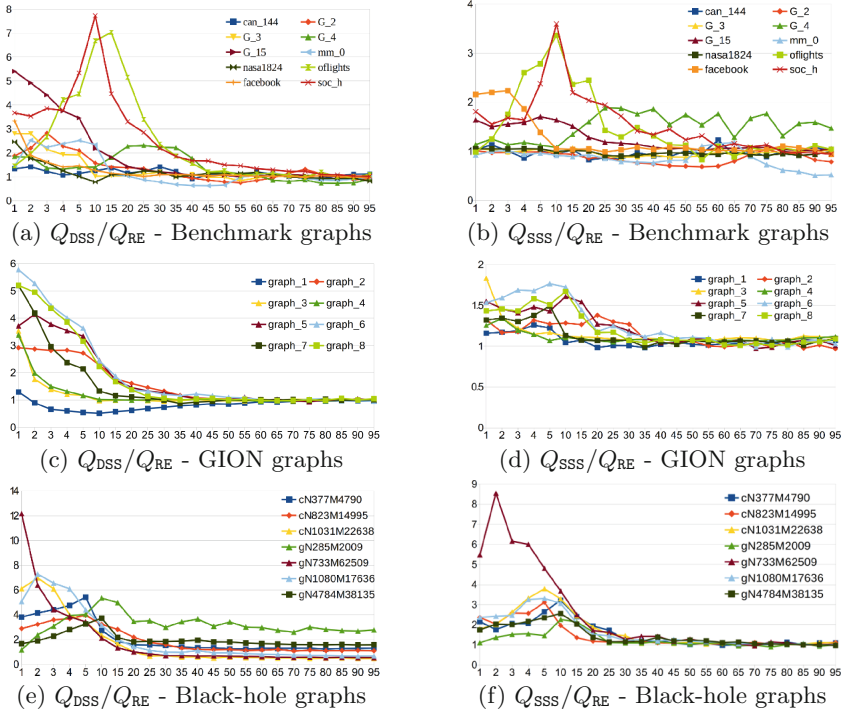


Fig. 7. Proxy/quality ratio for DSS and SSS, for (a) and (b) defacto-benchmark graphs, (c) and (d) GION graphs, (e) and (f) black hole graphs. The y -axis shows the quality ratio. (Color figure online)

40%, there are fluctuations between SSS and DSS. For grid-like graphs, the DSS and SSS proxies give better average proxy measures than RE proxies for relative density less than 20%. For relative density greater than 35%, RE proxies improve. For scale-free graphs, DSS and SSS outperformed when relative density is under 80%.

Figure 8(b) shows the ratio of the quality average between DSS over RE and SSS over RE. Overall, the quality ratios decline when relative density increase. The ratios are good from 1.2 to 3 times better for relative density up to 20%. For both types of graphs, DSS gives best quality, then SSS comes second.

4. **Deterministic spectral sparsification is better.** We compared the average of quality metrics for DSS, SSS and RE sparsification. Figure 9 shows the average quality values for the three data sets. As expected, average values increase when the relative density increases. Note that DSS gives the best average and SSS is the second best.

Figure 10 shows the quality ratios Q_{DSS}/Q_{RE} and Q_{SSS}/Q_{RE} for all the data sets. Again, DSS gives an overall larger improvement over RE than SSS. The improvement of DSS over RE is good when relative density is less than 35%; SSS shows in improvement over RE as well, but it is not so dramatic.

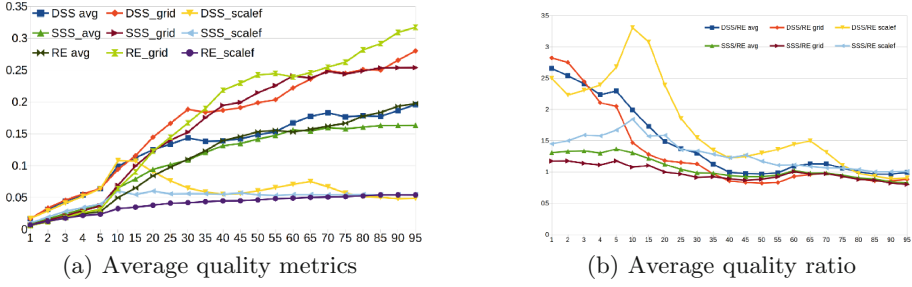


Fig. 8. Comparison of proxy quality metrics of defacto-benchmark graphs: (1) Average quality measures, (2) Average of quality ratio. The values are computed by graph types for scale-free graphs (`_scalef`), grid-like (`_grid`) graphs, and overall (`_avg`). Q_{DSS}/Q_{RE} .

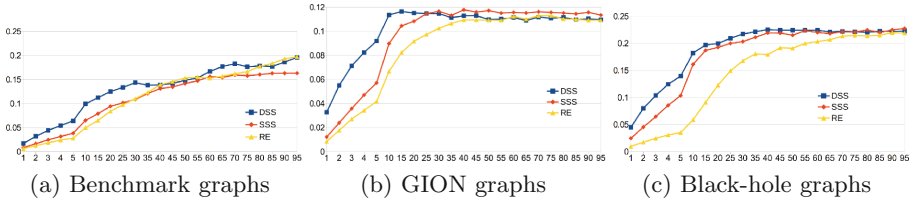


Fig. 9. Average quality metrics of DSS, SSS and RE over all data sets.

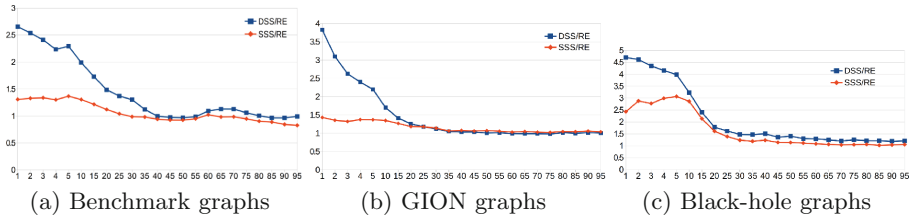


Fig. 10. Quality ratios of DSS/RE and SSS/RE over all data sets.

When relative density is beyond 35%, the ratio becomes small (close to 1) or even becomes smaller than 1. Further note from Fig. 10(a)–(c) that DSS and SSS give better quality ratios for black-hole graphs than for GION graphs and defacto-benchmark graphs.

- Quality results are consistent across different layout algorithms.** The results reported above use FM^3 for layout. However, we found that results using other layout algorithms were very similar. We measured the quality ratios using FM^3 , *Fast*, *Nice* and *NoTwist* layouts from OGDF. For example, Fig. 11 shows the quality ratio of DSS. As depicted from the graphs, the improvement of DSS over RE is consistent across different layout algorithms. The differences in the ratios is very small.

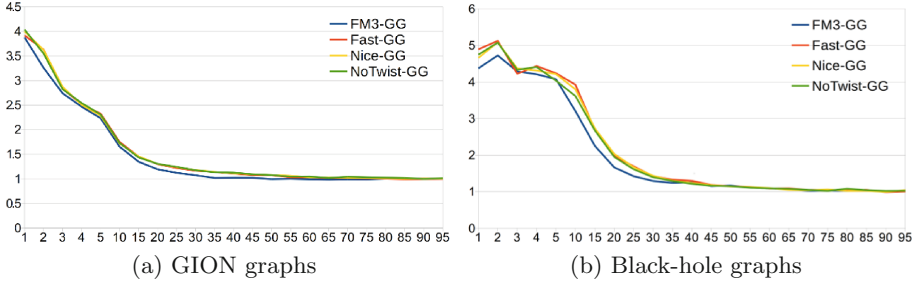


Fig. 11. Comparison of average quality ratio of DSS over RE between FM³, Fast, Nice and NoTwist layouts. The y-axis shows the average quality ratio Q_{DSS}/Q_{RE} .

5.2 Runtime

Although the main purpose of our investigation was to evaluate the *effectiveness* of spectral sparsification, some remarks about runtime are in order.

Figure 12(a) illustrates runtimes. The x-axis shows the number of edges, and the y-axis shows the computation time in minutes. Figure 12(b) shows the amount of time for (parallel) computing resistance values. The x-axis shows the number of edges, and the y-axis shows the computation time in seconds.

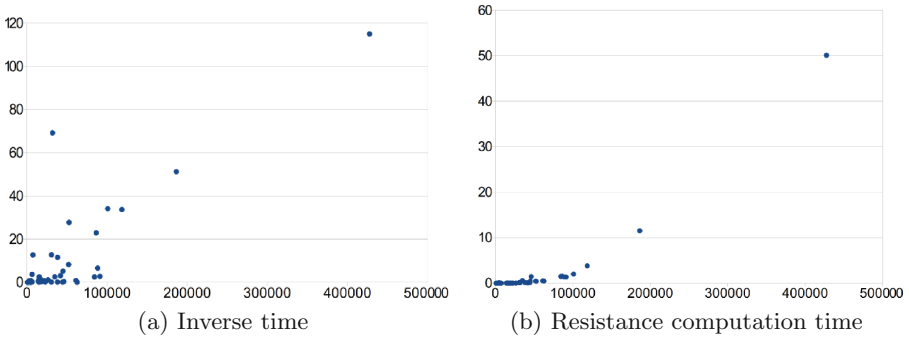


Fig. 12. The running time of computing Moore-Penrose inverse (in minutes) and resistance values of all edges (in seconds).

The dominant part of runtime is the computation of the Moore-Penrose inverse (and thus effective resistance); for this we used standard software [13]. For the defacto-benchmark graphs, computing the Moore-Penrose inverse takes 10.42min on average. Graph *can_144* takes minimum time for the Moore-Penrose inverse calculation (0.0003 min), and graph *graph_3* takes the longest time (115 min).

6 Concluding Remarks

This paper describes the first empirical study of the application of spectral sparsification in graph visualization.

Our experiments suggest that spectral sparsification approaches (DSS and SSS) are better than random edge approach. Further, the results suggest some guidelines for using spectral sparsification:

- DSS works better than SSS in practice.
- DSS and SSS give better quality metrics for grid-like graphs than for scale-free graphs.
- For sparsifications with low relative density (1% to 20%), DSS and SSS are considerably better than edge sampling. For relative density larger than 35%, RE may be more practical, because it is simpler, faster, and produces similar results to DSS and SSS.

Future work includes the following:

- Improve the runtime of these methods. For example, Spielman and Srivastava [25] present a nearly-linear time algorithm that builds a data structure from which we can query the approximate effective resistance between any two vertices in a graph in $O(\log n)$ time. This would allow testing spectral sparsification for larger graphs.
- More extensive evaluation: our experiments compare spectral sparsification with random edge sampling, but not with the wide range of sampling strategies above. Further, extension to larger data sets would be desirable.
- In our experiments, quality is measured using an objective shape-based metric. It would be useful to measure quality subjectively as well, using graph visualization experts as subjects in an HCI-style experiment.

References

1. Batson, J.D., Spielman, D.A., Srivastava, N., Teng, S.: Spectral sparsification of graphs: theory and algorithms. *Commun. ACM* **56**(8), 87–94 (2013)
2. Ben-Israel, A., Greville, T.N.: *Generalized Inverses: Theory and Applications*, vol. 15. Springer Science & Business Media, New York (2003)
3. Benczúr, A.A., Karger, D.R.: Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, Philadelphia, Pennsylvania, USA, 22–24 May 1996, pp. 47–55 (1996)
4. Chimani, M., Gutwenger, C., Jünger, M., Klau, G.W., Klein, K., Mutzel, P.: *The Open Graph Drawing Framework (OGDF)*. CRC Press, Boca Raton (2012)
5. Chung, F.: *Spectral Graph Theory*. American Maths Society (1997)
6. Davis, T.A., Hu, Y.: The University of Florida sparse matrix collection. *ACM Trans. Math. Softw.* **38**(1), 1:1–1:25 (2011)
7. Eades, P., Hong, S., Nguyen, A., Klein, K.: Shape-based quality metrics for large graph visualization. *J. Graph Algorithms Appl.* **21**(1), 29–53 (2017). <https://doi.org/10.7155/jgaa.00405>

8. Eades, P., Hong, S., Nguyen, A., Klein, K.: Shape-based quality metrics for large graph visualization. *J. Graph Algorithms Appl.* **21**(1), 29–53 (2017)
9. Gjoka, M., Kurant, M., Butts, C.T., Markopoulou, A.: Walking in facebook: a case study of unbiased sampling of OSNs. In: *Proceedings of the 29th Conference on Information Communications, INFOCOM 2010* pp. 2498–2506. IEEE Press, Piscataway (2010)
10. Godsil, C.D., Royle, G.F.: *Algebraic Graph Theory*. Graduate Texts in Mathematics. Springer, New York (2001). <https://doi.org/10.1007/978-1-4613-0163-9>
11. Gross, J., Yellen, J.: *Handbook of Graph Theory*. CRC Press, Boca Raton (2004)
12. Hachul, S., Jünger, M.: Drawing large graphs with a potential-field-based multilevel algorithm. In: Pach, J. (ed.) *GD 2004*. LNCS, vol. 3383, pp. 285–295. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31843-9_29
13. Hare, J.S., Samangooei, S., Dupplaw, D.: OpenIMAJ and ImageTerrier: Java libraries and tools for scalable multimedia analysis and indexing of images. In: *Proceedings of the 19th International Conference on Multimedia 2011*, pp. 691–694 (2011)
14. Hu, P., Lau, W.C.: A survey and taxonomy of graph sampling. *CoRR abs/1308.5865* (2013)
15. Hu, Y., Shi, L.: Visualizing large graphs. *WIREs Comput. Stat.* **7**(2), 115–136 (2015). <https://doi.org/10.1002/wics.1343>
16. Leskovec, J., Faloutsos, C.: Sampling from large graphs. In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 631–636. ACM (2006)
17. Lovász, L.: Random walks on graphs: a survey. In: Miklós, D., Sós, V.T., Szőnyi, T. (eds.) *Combinatorics, Paul Erdős is Eighty*, vol. 2, pp. 353–398. János Bolyai Mathematical Society (1996)
18. Marner, M.R., Smith, R.T., Thomas, B.H., Klein, K., Eades, P., Hong, S.-H.: GION: interactively untangling large graphs on wall-sized displays. In: Duncan, C., Symvonis, A. (eds.) *GD 2014*. LNCS, vol. 8871, pp. 113–124. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45803-7_10
19. Morstatter, F., Pfeffer, J., Liu, H., Carley, K.: Is the sample good enough? Comparing data from Twitter’s streaming API with Twitter’s firehose, pp. 400–408. AAAI press (2013)
20. Nguyen, Q.H., Hong, S.H., Eades, P., Meidiana, A.: Proxy graph: visual quality metrics of big graph sampling. *IEEE Trans. Visual Comput. Graphics* **23**(6), 1600–1611 (2017)
21. Nguyen, Q., Eades, P., Hong, S.-H.: On the faithfulness of graph visualizations. In: Didimo, W., Patrignani, M. (eds.) *GD 2012*. LNCS, vol. 7704, pp. 566–568. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36763-2_55
22. Nguyen, Q.H., Eades, P., Hong, S.: On the faithfulness of graph visualizations. In: *IEEE Pacific Visualization Symposium, PacificVis 2013*, 27 February–1 March 2013, Sydney, NSW, Australia, pp. 209–216 (2013). <https://doi.org/10.1109/PacificVis.2013.6596147>
23. Rafiei, D., Curial, S.: Effectively visualizing large networks through sampling. In: *16th IEEE Visualization Conference, VIS 2005*, Minneapolis, MN, USA, 23–28 October 2005, pp. 375–382 (2005)
24. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: *AAAI (2015)*. <http://networkrepository.com>
25. Spielman, D.A., Srivastava, N.: Graph sparsification by effective resistances. *CoRR abs/0803.0929* (2008). <http://arxiv.org/abs/0803.0929>

26. Spielman, D.A., Teng, S.: Spectral sparsification of graphs. *SIAM J. Comput.* **40**(4), 981–1025 (2011)
27. Toussaint, G.T.: The relative neighbourhood graph of a finite planar set. *Pattern Recogn.* **12**(4), 261–268 (1980). [https://doi.org/10.1016/0031-3203\(80\)90066-7](https://doi.org/10.1016/0031-3203(80)90066-7)
28. Von Luxburg, U.: A tutorial on spectral clustering. *Stat. Comput.* **17**(4), 395–416 (2007)
29. Wu, Y., Cao, N., Archambault, D., Shen, Q., Qu, H., Cui, W.: Evaluation of graph sampling: a visualization perspective. *IEEE Trans. Visual Comput. Graphics* **23**(1), 401–410 (2017)

Revisited Experimental Comparison of Node-Link and Matrix Representations

Mershack Okoe¹, Radu Jianu²(✉), and Stephen Kobourov³

¹ Department of Computer Science, Florida International University, Miami, USA
mokoe001@cis.fiu.edu

² giCentre, City, University of London, London, UK
radu.jianu@city.ac.uk

³ Department of Computer Science, University of Arizona, Tucson, USA
kobourov@cs.arizona.edu

Abstract. Visualizing network data is applicable in domains such as biology, engineering, and social sciences. We report the results of a study comparing the effectiveness of the two primary techniques for showing network data: node-link diagrams and adjacency matrices. Specifically, an evaluation with a large number of online participants revealed statistically significant differences between the two visualizations. Our work adds to existing research in several ways. First, we explore a broad spectrum of network tasks, many of which had not been previously evaluated. Second, our study uses a large dataset, typical of many real-life networks not explored by previous studies. Third, we leverage crowdsourcing to evaluate many tasks with many participants.

1 Introduction

Visualizing network data is known to benefit a wide range of domains, including biology, engineering, and social sciences [54]. The data visualization community has proposed many approaches to visual network exploration. By comparison, the body of work that evaluates the ability of such methods to support data-reading tasks is limited. We describe the results of a comparative evaluation of the two most popular ways of visualizing networks: node-link diagrams (NL) and adjacency matrices (AM). Specifically, we consider two interactive visualizations (NL and AM), using a crowdsourced, between-subject methodology, with 557 distinct online users, 14 evaluated tasks, and 1 real-world dataset; see Fig. 1.

Several earlier studies compare NL and AM visualizations on specific classes of networks and using a variety of tasks [21, 22, 30, 36]. They show that the effectiveness of the visualization depends heavily on the properties of the given dataset and the given data-reading tasks. For example, Ghoniem *et al.*'s seminal evaluation [21] found that the two visualizations' ability to support specific tasks depends on the size and density of the network. Similarly, it is reasonable to hypothesize that there might be differences depending on the structure of the network (e.g., clustered networks, small-world networks). Thus exploring

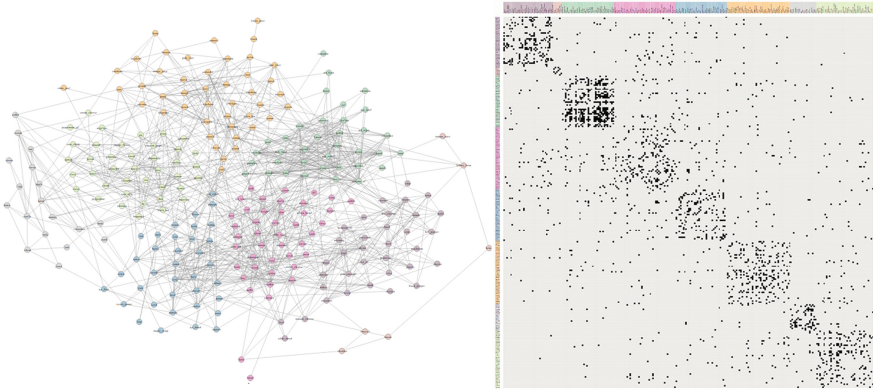


Fig. 1. Evaluated visualizations: node-link diagram and adjacency matrix.

the effectiveness of NL and AM visualizations on different types of graphs, and using a broader spectrum of tasks, seems worthwhile.

Our study uses one real-world, scale-free dataset of 258 nodes and 1090 edges. This makes our dataset different in structure and larger than previously evaluated networks. For example, Ghoniem *et al.* evaluated random networks that were about 2.5 times smaller, albeit somewhat denser. We argue (in Sect. 3) that our chosen dataset is worth studying as it exemplifies a large class of networks that occur in real applications.

More recently, networks are used to solve increasingly complex problems and as a result, there is an expanding range of tasks that are relevant in real applications and which are of interest to the visualization community. Our study evaluates many tasks (14), carefully chosen to span multiple task taxonomies [4, 32]. Many of these tasks were not previously investigated in the context of NL and AM representations.

Given the caveat that these results apply to the specific underlying network and the specific implementations of NL and AM visualizations, some of our results confirm prior observations in similar settings, while others are new. NL outperforms AM for questions about graph topology (e.g., “Select all neighbors of node,” “Is a highlighted node connected to a named node?”). Of 10 such tasks, participants who used the node-link diagram were more accurate in 5 and less accurate in 2. NL and AM give similar results for 4 tasks which tested the ability of the participants to identify and compare node groups or clusters, except one instance in which AM outperforms NL. Finally, NL and AM provide similar results on 2 memorability tasks. The full results are shown in Fig. 4.

2 Related Work

Considerable effort has been expended on optimizing NL and AM visualizations to remove clutter, increase the saliency of visual patterns, and support data

reading tasks [54]. NL, AM, and slight variations thereof have long been used in practice to support analyses of data in a broad range of domains, including proteomic data [8, 28, 29, 50], brain connectivity data [3], social-networks [53], and engineering [49].

Static visual encodings were augmented by interaction to support the exploration and analysis of large and intricate datasets typical of real-life applications. Interactive systems that visualize complex relational data use NL [6, 9, 50], and AM [7, 10–12, 17, 20, 45, 51]. We reviewed such systems to determine common interactions and included them in our evaluated visualizations.

While the two types of visualizations have been used broadly for a long time, studying how people parse them visually and which visualization method better supports specific tasks and datasets, is ongoing. For example, studies by Purchase *et al.* [40, 41, 55] consider how node-link layouts impact data readability, eye-tracking research by Huang *et al.* reveal visual patterns and measure the cognitive load associated with network exploration [26, 27]. More recently Jianu *et al.* and Saket *et al.* consider the performance of node-link diagrams with overlaid group information [28, 48].

Our work is one in a series of studies that compare NL and AM representations. Ghoniem *et al.* [21] evaluated the two approaches on seven connectivity and counting tasks, using interactive visualizations (e.g., node can be selected and highlighted). Synthetic graphs of three sizes (20, 50, 100 nodes) and three densities (0.2, 0.4, and 0.6) were used. The authors found that for small sparse graphs, NL was better in connectivity tasks, but that for large and dense graphs, AM outperformed NL for all tasks. Similarly, Keller *et al.* [30] evaluated six tasks on three real-life networks of varying small sizes (8, 22, 50) and three densities (unspecified, 0.2, 0.5). Using both static and interactive variants of NL and AM, Abuthawabeh *et al.* found that the participants were equally able to detect structure in graphs representing code dependencies [1]. Alper *et al.* found that in tasks involving the comparison of weighted graphs, matrices outperform node-link diagrams [3]. Finally, Christensen *et al.* [16] evaluated matrix quilts in addition to NL and AM in a smaller scale study.

Our study adds to what is already known in several ways. First, we explore a significantly broader range of tasks than earlier studies. These were carefully selected to cover the graph task taxonomy of Lee *et al.* [32] and the general taxonomy of visualization tasks by Amar *et al.* [4]. We also considered the task taxonomies for simple graphs [32], clustered graphs [47], and more generally for visualization tasks [4, 52], which have been found to be useful in guiding research and informing user study task choices [28, 48]. Second, our study uses a large real-world network, typical of many scale-free networks that arise in practical applications. Finally, unlike previous studies, we leverage crowdsourcing, via Amazon’s Mechanical Turk, to evaluate many tasks with many participants.

Note that Mechanical Turk provides access to a diverse participant population [31, 33], and is considered a valid platform for evaluation in general [31, 38], as well as specifically in the context of visualization studies [23]. Many recent visualization studies are crowdsourced [14, 15, 28, 35, 43] and specific platforms

for online evaluations are developed, including GraphUnit designed for online evaluation of network visualizations [37].

3 Study Design

3.1 Stimuli: Data

We evaluated a single network with 258 nodes and 1090 edges, representing cooking ingredients connected by edges when frequently used together in recipes. The density of the network was 0.016 (computed as $\#edges/\#nodes^2$). This network had been explored previously by Ahn *et al.* [2]. In its original form, the network is larger (381 nodes) but we reduced it slightly to ensure it could be visualized smoothly in a browser. We did so by removing disconnected components and low-weight edges. Evaluating a single dataset allowed us to cover a broad spectrum of tasks while keeping the size of the study manageable, but naturally, this choice has several limitations, discussed in Sect. 5.

Rationale: Our motivation for choosing our network was three-fold. First, it is *different than those evaluated already*. Our network is 2.5 and 5 times larger than those evaluated by Ghoniem *et al.* and Keller *et al.* Second, our network was chosen as a *representative of several types of real-world networks*. Specifically, we reviewed 17 relational datasets (e.g., trade exchanges between countries, the Les Miserable dataset, TVCG paper co-authorships, protein-interaction networks). We selected one from this set that was representative in terms of structure and density, while at the same time sufficiently small to be evaluated in a browser. Our network has about 4 times more edges than nodes. This was close to the average edge/node ratio in the 17 networks we reviewed and representative of many networks commonly found in practice [34]. Third, we believe a dataset revolving around cooking ingredients would have a *greater appeal to participants*. Ingredients were shown as node labels and several tasks referred to ingredients by name. Relatable, concrete dataset may help users understand tasks better [5].

3.2 Stimuli: Visual Encoding

We evaluated two visual encodings: a node-link diagram (NL) drawn using the neato algorithm from graphviz [18], and an adjacency matrix (AM), sorted to reveal clusters using the barycenter algorithm available in the Reorder.js library [20]. We clustered the network using modularity clustering from GMap [25] and encoded this information in the two visual representations using color, as shown in Fig. 1. Both visualizations were developed using the D3 web-library.

Rationale: The neato algorithm is provided in popular layout tools such as graphviz and frequently part of NL evaluations [21, 28]. We ordered our AM to reveal structure, as we considered this more representative of how matrices are used in practice, unlike Ghoniem *et al.* [21], who used a lexicographical order.

3.3 Stimuli: Interactions

Both visualizations support panning and zooming, using the mouse-wheel. Multiple nodes can be selected by clicking on them, and deselected with an additional click. Selecting a node in NL colors both the node and its outgoing edges in purple. Selections in AM operate on node labels but change the color of the corresponding node’s row or column. Similarly, node mouse-over in NL turns the node and its edges green and shows the node label via tooltips. Node mouse-over in AM colors the row or column. Note that for both node selection and node mouse-over in AM, if a row (column) is colored the complementary column (row) is not. We chose this approach since both Ghoniem *et al.* and Okoe *et al.* mention that multiple markings for the same node can confuse users [21, 28].

To select a node as the answer to a task, the participants double-click it. This marks the node with a thick black contour. In both NL and AM this marking was restricted to nodes and labels, without extending to edges or rows/columns. The participants could also deselect an answer by double-clicking it again.

Similar interactions apply to edge selection: An edge mouse-over in NL turns the edge green, and if clicked it is selected and so turns purple. In AM, hovering over an edge-cell highlights its corresponding row and column in green, and clicking it selects the edge.

Rationale: We chose to evaluate interactive visualizations as interactivity is typical in real-world applications. Previous studies, such as those of Ghoniem *et al.* or Keller *et al.*, also used basic interactions for the same reason. Interactivity can significantly change the effectiveness of a visual encoding, however, and a careful choice of interactive techniques is warranted.

Our goal was to use interactions that are *ecologically valid* (i.e., representative of interactions typical of NL or AM visualizations) and *fair* (i.e., providing similar functionality and power in both visualizations). To this end, we reviewed 9 systems for network visualization (e.g., Gephi [9], Cytoscape [50], Tulip [6]), 12 network evaluation papers (e.g., Ghoniem *et al.* [21], Keller *et al.* [30], Okoe *et al.* [36]) and 6 systems and papers for adjacency matrices (e.g., ZAME [19], TimeMatrix [56], work by Perin *et al.* [39], work by Henry *et al.* [24]). We cataloged the interactions described or available in these systems, as well as their particular implementation, and then selected the set of most common interactions.

This resulted in a set of interactions that both overlapped and differed slightly from those implemented in previous studies. Overlapping interactions were described above. New interactions included zooming and panning, which was required to solve some of the tasks. We believe the addition of zooming and panning is valuable since such basic navigation is an integral part of real-life systems. Our node-link diagrams also allowed users to move nodes, an interaction that can be used to disambiguate cases in which nodes or edges overlap, and is ubiquitous in NL systems. This interaction does not have an equivalent in AM but is also not necessary as rows and columns are evenly spaced.

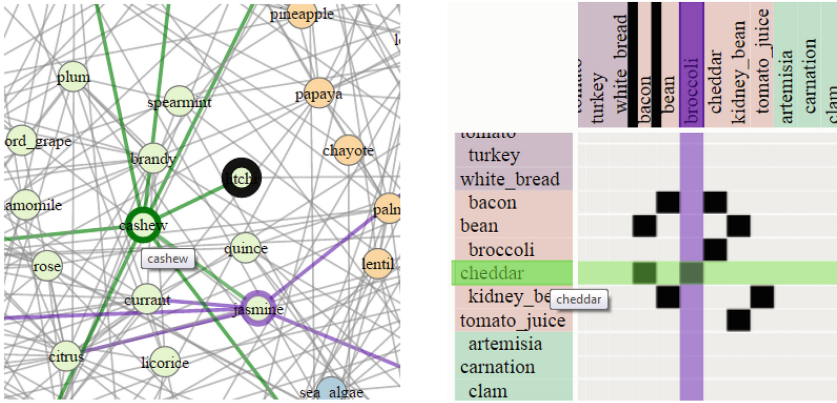


Fig. 2. Participants mouse-over nodes to highlight them (green) and click on nodes to select them (purple). Designating a node as the answer for a task answer is accomplished via a double-click, which draws a black contour around the node. (Color figure online)

3.4 Tasks

We evaluated the 14 tasks described in Table 1. Participants solved multiple repeats (generally 5 or 10) of each task. Task repeats were selected manually on the network so as to cover multiple levels of complexity. For example, our repeats included nodes with both low and large degrees (e.g., $T1, T2$), short and long paths (e.g., $T10, T13$), or nodes with few and many neighbors (e.g., $T4$).

Three of our tasks warrant a more detailed discussion. We included two memorability tasks, ($T11, T14$). The former tested the ability of participants to recall data they had looked for or accessed at an earlier time, and is similar to memorability tasks evaluated by Saket *et al.* [46]. The latter tested the ability of participants to recognize visual configurations they had viewed previously and is more similar to tasks used by Jianu *et al.* and Borkin *et al.* [14,28]. Both memorability tasks were based on questions that the participants had to answer early in their session (i.e., $T9$ in group 4, and $T12$ in group 5) to prime the participants with a particular piece of information or visual configuration. A few minutes later, after performing a set of other tasks (i.e., $T10$ in group 4, $T13$ in group 5), the participants were asked about the information from the earlier task. Finally, we added a path-estimation task ($T5$), which required the participants to estimate how far two nodes are, in terms of the shortest path between them. Timing constraints ensured that participants used perceptual mechanisms to give a best-guess response instead of “computing” the correct answer.

Rationale: Our overarching goal in selecting our tasks was to cover a wide spectrum of different and realistic network tasks. We selected tasks to cover the graph objects they provide answers about (i.e., nodes, edges, paths), as well as cover Lee *et al.*’s categories of graph-reading tasks, and Amar *et al.*’s general types of visualization tasks. Several of our tasks have been used before

but under slightly different conditions. Additionally, we included tasks that go beyond previous studies comparing NL and AM, such as tasks involving clusters. We also included memorability tasks as they are a topic of growing interest in the visualization community [14, 46]. We also hypothesized there would be differences between the two visualizations in this respect. We included a path-estimation task [28], as it is a good representative of the “Overview” category of graph tasks, and underlies perceptual queries that users make on relational data.

3.5 Hypotheses

Based on previous results by Ghoniem *et al.* [21], Keller *et al.* [30], Okoe *et al.* [37], Jianu *et al.* [28], and Saket *et al.* [48] we devised the null hypotheses:

H1: There is no statistically significant difference in time and accuracy performance between using NL and AM for tasks involving the retrieval of information about nodes and direct connectivity ($T1, T2, T4, T9, T12$).

H2: There is no statistically significant difference in time and accuracy performance between using NL and AM for connectivity and accessibility tasks involving paths of length greater than two ($T5, T10, T13$).

H3: There is no statistically significant difference in time and accuracy performance between using NL and AM on group tasks ($T3, T6, T7, T8$).

H4: There is no statistically significant difference in memorability between using NL and AM.

We expected H1 to hold and H2 not to hold. We also thought H3 would hold, except for estimating group interconnectivity ($T6$), since estimating the number of non-overlapping dots in a square (AM) should be easier than estimating overlapping edges in an irregular 2D area (NL). Finally, we anticipated memorability would be higher in node-link diagrams due to its more distinguishable features.

3.6 Design

We used a between-subjects experiment with two conditions. We divided our 14 task types into 5 experimental groups, as shown in Table 1, and we evaluated each group separately. Each participant was allowed to participate in a single group and used just one of the two visualizations. We assigned participants to groups and conditions in a round-robin fashion. We aimed to collect data from around 50 participants per condition. As some participants did not complete the study, the total number of participants for whom we collected data varies slightly between conditions. All tasks were timed as shown in Table 1, with time limits determined experimentally through a pilot-study and chosen to allow most participants to complete the tasks, while moving the study along.

Rationale: Between-subject experiments are frequently used in the visualization community [13, 28, 31, 35, 42, 48, 57]. One advantage of this design is the absence of learning effects between evaluated conditions. A disadvantage is the need

Table 1. Tasks: the columns describe (i) the task, (ii) targeted network element, (iii-iv) task categories in Lee *et al.*'s and Amar *et al.*'s taxonomies, (v) group number the task was evaluated in, (vi) number of instances of this task type, (vii) task time limit (sec).

Task	Target	Task tax. [32]	Task tax. [4]	Group	#Repeats	Time
1. Given two highlighted nodes, select the one with the larger degree	node	Topology (adjacency)	Retrieve value, Sort	1	10	15
2. Given a highlighted node, select all its neighbors	edge	Topology (adjacency, accessibility)	Retrieve value, Filter	1	10	25
3. Given two clusters of highlighted nodes, which one is more interconnected?	clusters, cliques	Overview (connectivity)	Filter, Sort, Cluster	1	10	10
4. Given two highlighted nodes, select all of the common neighbors	edge	Topology (shared neighbor)	Retrieve value, Filter	2	10	30
5. Given two pairs of highlighted nodes (red and blue) and limited time, estimate which pair is closer in terms of graph topology?	path, edge	Overview (connectivity)	Derive value, Sort	2	10	10
6. How many clusters are there in the visualization? *clusters shown via color (Sect. 3.2)	clusters	Overview (connectivity)	Derive value	3	1	10
7. Given two groups of highlighted nodes (e.g., red and blue) and limited time, estimate which group is larger	clusters	Attribute based	Filter, Sort, Derive value, Correlate	3	10	10
8. Given two highlighted nodes decide whether they belong to the same cluster *clusters shown via color (Sect. 3.2)	clusters, nodes	Attribute based	Cluster, Filter	3	10	10
9. Given one highlighted node and one named node, are they connected?	edge	Topology (adjacency)	Retrieve value	4	5	20
10. Given two highlighted nodes, how long is the shortest path between them?	path, edge	Topology (connectivity)	Retrieve value, Derived value, filter	4	5	60
11. Memorability: After spending several minutes on task 10, can participants remember the answers they gave to task 9, without access to the visualization?		See Sect. 3.4	See Sect. 3.4	4	5	unlim
12. Given two highlighted nodes and three named ones, which of the named nodes is connected to both highlighted nodes? (exemplified in Fig. 3)	edge	Topology (shared neighbor)	Retrieve value, Filter	5	5	60
13. Given a selected node, how many nodes are within two edges' reach?	edge	Topology (accessibility)	Retrieve value, Derive value, Filter	5	5	60
14. Memorability: After spending several minutes on tasks 13, can participants remember (i.e., select) which nodes were highlighted as part of task 12, if showed the visualization with the answers they gave to task 13 highlighted?		**See paper body	**See paper body	5	5	unlim

for large numbers of participants, which is easily mitigated in a crowdsourced setting. Moreover, between-subjects designs are quicker (since only one condition is evaluated at a time) and online participants prefer shorter studies.

We divided the tasks into groups for the same reason. Having each participant evaluate all tasks would have resulted in excessively long sessions that participants would have found tiring. Having participants solve only subsets of tasks allowed us to reduce their time commitment. We used estimated task completion times to group tasks, aiming for an expected duration of about 15 min.

We aimed for 50 participants per condition, matching the numbers used in earlier crowdsourced studies [15, 28]. We decided to enforce short time-limits in order to limit and make uniform the total session duration across participants.

3.7 Procedure

We used Amazon’s Mechanical Turk (MTurk) to crowdsource our study to a broad population. To account for variations in participant demographics during the day, we published study batches throughout the day. We ran conditions in parallel and directed incoming participants to them using a round-robin assignment, to ensure that the two conditions sampled participants from the same populations. The demographics of MTurk users are reported by Ross *et al.* [44].

Each incoming participant was first presented with an introduction to the study, dataset, the visualization they would see and use, and the tasks they would perform. Each task was exemplified in the introduction, as shown in Fig. 2. Since our interactions relied on color, participants were administered a color-blindness test. Next came a training session which involved solving two instances of each type of task in their assigned group. During the training session the participants could check the correctness of their answers.

Finally, the participants were lead to the main part of the study. In the main part of the study, task instances of each type in an assigned group were shown to the participants. For example, since group 1 involved three distinct task types, participants assigned to it solved three consecutive sections of ten task-instances each. At the end, we asked the participants for comments.

We used GraphUnit [37] to create the study, deploy it, and collect data. Visualizations were shown on the left, while task instructions and answer widgets were shown on the right. Depending on each task, users answered by selecting nodes or by using interactive widgets (e.g., text-boxes, check-boxes). Time limits were enforced by showing a count-down timer and hiding the visualization once the counter expired. To increase the chances of collecting clean data we awarded a bonus to the best result in each group and told participants that two of the task-instances were control tasks easy enough for anyone to solve.

4 Results

Our results are summarized in Fig. 4. By and large, they show that node-link diagrams were better for most types of connectivity tasks ($T1$, $T2$, $T4$, $T5$, $T9$,

T10, T13) thereby invalidating both H1 and H2. The fact that H1 does not hold is surprising given previous results. Performance on group tasks was generally comparable with the two visualizations, as hypothesized (H3), though we found that the AM was better for estimating the number of clusters rather than their interconnectivity. Finally, NL supported memorability tasks better (invalidating

Groups	Condition	User Size	Valid Data
1	NL	65	63
	AM	62	58
2	NL	58	54
	AM	53	50
3	NL	55	53
	AM	55	52
4	NL	52	50
	AM	53	50
5	NL	54	52
	AM	50	47

Fig. 3. Number of participants in each task group per condition and the number of valid submissions used after data cleaning.

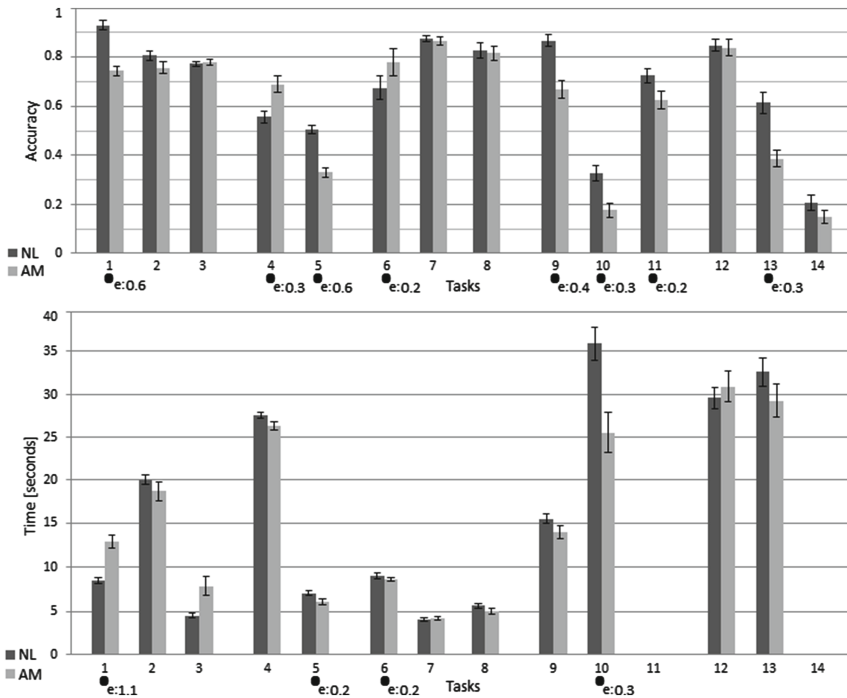


Fig. 4. Results: accuracy and time. Error bars show one standard error. Statistically significant results and effect sizes are also marked. Tasks 14, 11 had no time limits.

H4). In particular, NL users outperformed AM users when recalling previously used data (*T11*).

Data Processing: We collected data from 557 individual participants distributed across task groups and conditions as shown in Fig. 3. We removed a total of 28 responses from participants who spent an average of 2s per task and had accuracy in the bottom 10 percentile. We considered these likely to be random responses by participants attempting to game the study.

To compute the accuracy of node selections (*T1*, *T2*, *T4*), we used the formula $Acc = (||PS \cap TA||)/||TA||$, where *PS* is the participant’s selection and *TA* is the true answer. To compute answers for tasks involving numeric answers (*T6*, *T10*, *T13*) we used the formula $Acc = \max(0, 1 - ||PA - TA||/|TA|)$, where *PA* is the participant’s answer and *TA* is the true answer. For other tasks we gave a 1 to correct answers, and a 0 to incorrect answers. Since each task type was represented in the study by several repeats, we averaged the accuracies of a task’s individual repeats into an accuracy for the task as a whole.

Statistical Analysis: If the data is normally distributed (determined via a Shapiro-Wilk test) we use a t-test analysis between conditions to determine if the observed differences are significant. Otherwise we use a Wilcoxon-Rank-Sum test. We indicate statistically significant differences and effect sizes in Fig. 4.

5 Discussion

Based on the quantitative results and our own interactions with the visualizations, we believe the results can be explained by several factors.

First, NL can be more compact than AM since their layout fully leverages the 2D area, while matrices are constrained to two 1D linear node orders. Matrices favor dense networks (as number of edges increases, matrix size remains constant) but not sparse ones (empty matrices are as large as a dense ones). Instead, sparse NL diagrams can be packed tightly. At the extreme, an empty network can be shown without loss in readability using NL in a $\sqrt{N} \times \sqrt{N}$ square. The same empty network would require a $N \times N$ square in an AM. Thus, as networks grow larger but not necessarily denser, AM may incur an increasing navigation cost. Concretely, our NL diagrams required less zooming for nodes to become legible and selected accurately. This could explain the differences in *T1*.

Second, NL draw a node’s glyph and connections together. Thus, once a label is spotted, from it, its outgoing edges can be traced to other nodes and their labels. Moreover, the presence of the edge aids this tracing. Instead, matrices show node information and edge information separately. Finding the endpoints of an edge involves two potentially long visual-traces along the horizontal and vertical axes. Similarly, finding an edge of an identified node involves a horizontal or vertical search. This could be one of the reasons for the large effect in *T9*. However, this described behavior is only hypothesized and yet to be demonstrated.

Third, Ghoniem *et al.* found that AM performs poorly on tasks involving long paths [21], and our results on $T10$ and $T13$ confirm this. Interestingly, the average time of participants performing path tasks ($T10$) in AM is significantly shorter than that for NL. However, we found that this is due to many AM users giving up on solving the task altogether early on. Moreover, NL layouts aim to place nodes so that their network distance matches their embedded distance. While matrices can also order rows and columns, they are constrained by the use of a single dimension. This could explain the results of $T5$: when one pair of nodes were in the same cluster and the other not, comparing their topological proximity was possible in both visualizations, but in all other cases NL outperforms AM.

Matrices eliminate occlusion and ambiguity problems. In NL diagrams it is sometimes difficult to tell if an edge connects to a node or passes through it, but this is not the case in AMs. Moreover, many tasks that involve visual searches in unconstrained 2D space with NL, are easier with AM. For example, finding a node in an AM involves a linear scan in a list of labels. Counting nodes with certain properties can also be done sequentially by moving through the matrix's headers. Such tasks are difficult in NL diagrams as users have to search a 2D space and keep track of already visited nodes. This may account for $T4$, where AM outperforms NL: participants could systematically scan two selected AM node-rows and identify the columns where both rows had an edge.

Limitations: Several earlier studies comparing NL and AM considered the effects of network size and density [22,30]. While we recognize the value of this approach, this was beyond the scope of our current study. Instead, we aimed to understand how the two visualizations support a more complete range of tasks (14 versus previously 7 and 6) in a network that is representative of real-world networks in size and structure. It is unclear whether our results would generalize to real-world networks that are significantly larger or denser but our work does provide additional experimental data for a network unlike those evaluated earlier.

We use one type of network and a single instance thereof. This is a methodological drawback which we accepted, due to the overhead associated with preparing multiple appropriate real-world networks for evaluation and phrasing participant instructions using the semantics of different networks. While the limitations of this approach are non-trivial, we attempted to balance them by using multiple task-repeats of the same type and focusing on different parts of the network.

The density of our network was significantly lower than [21,30]. However, Melancon points out that large real-world networks with high densities are rare [34]. He argues that the edge-to-node ratio is a better indicator for density in real-world networks as it is less sensitive to the number of nodes. Indeed, only 1 of the 17 networks we considered, and 3 of the 19 networks Melancon considered had densities higher than 0.2. In 3 of these 4 cases, these dense networks were also the smallest in terms of number of nodes.

As in recent studies, we evaluate interactive visualizations. Given the different visual encoding in NL and AM it is difficult to ensure that all interactions are fair

to both visualizations. To alleviate this concern we relied on a detailed review of the NL and AM literature, and selected the most common interactions and their implementations (see Sect. 3.3). This ensured, at least to some degree, that we evaluated the interactive visualizations as they appear in practice.

Crowdsourced studies have known inherent limitations (e.g., difficulty controlling the experimental setup and verifying what participants do). By and large, however, crowdsourcing studies replicate prior controlled lab studies [23].

6 Conclusions

We presented the results of a crowdsourced evaluation of NL and AM network visualizations. Our study involved 557 online participants who used interactive versions of the two encodings, to answer 14 varied types of questions about a large network of 256 nodes and 1090 edges. We found that NL is better than AM for questions about network topology and connectivity, and comparable for group and memorability tasks, and therefore a better choice for visualizing datasets similar to the one we evaluated, provided a similar interaction set.

References

1. Abuthawabeh, A., Beck, F., Zeckzer, D., Diehl, S.: Finding structures in multi-type code couplings with node-link and matrix visualizations. In: 2013 First IEEE Working Conference on Software Visualization (VISSOFT), pp. 1–10. IEEE (2013)
2. Ahn, Y.Y., Ahnert, S.E., Bagrow, J.P., Barabási, A.L.: Flavor network and the principles of food pairing. *Scientific reports* 1 (2011)
3. Alper, B., Bach, B., Henry Riche, N., Isenberg, T., Fekete, J.D.: Weighted graph comparison techniques for brain connectivity analysis. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 483–492. ACM (2013)
4. Amar, R., Eagan, J., Stasko, J.: Low-level components of analytic activity in information visualization. In: 2005 IEEE Symposium on Information Visualization, INFOVIS 2005, pp. 111–117. IEEE (2005)
5. Archambault, D., Purchase, H.C., Hofffeld, T.: Evaluation in the Crowd: Crowdsourcing and Human-Centred Experiments. Springer, Cham (2017). <https://doi.org/10.1007/978-3-319-66435-4>
6. Auber, D.: Tulip: a huge graph visualization framework. In: Jünger, M., Mutzel, P. (eds.) *Graph Drawing Software*, pp. 105–126. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-642-18638-7_5
7. Bach, B., Pietriga, E., Fekete, J.D.: Visualizing dynamic networks with matrix cubes. In: Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems, pp. 877–886. ACM (2014)
8. Barsky, A., Gardy, J.L., Hancock, R.E., Munzner, T.: Cerebral: a cytoscape plugin for layout of and interaction with biological networks using subcellular localization annotation. *Bioinformatics* **23**(8), 1040–1042 (2007)
9. Bastian, M., Heymann, S., Jacomy, M., et al.: Gephi: an open source software for exploring and manipulating networks. *ICWSM* **8**, 361–362 (2009)

10. Behrisch, M., Davey, J., Fischer, F., Thonnard, O., Schreck, T., Keim, D., Kohlhammer, J.: Visual analysis of sets of heterogeneous matrices using projection-based distance functions and semantic zoom. In: *Computer Graphics Forum*, vol. 33, pp. 411–420. Wiley Online Library (2014)
11. Bezerianos, A., Dragicovic, P., Fekete, J.D., Bae, J., Watson, B.: Geneaquils: a system for exploring large genealogies. *IEEE Trans. Vis. Comput. Graph.* **16**(6), 1073–1081 (2010)
12. Blanch, R., Dautriche, R., Bisson, G.: Dendrogramix: a hybrid tree-matrix visualization technique to support interactive exploration of dendrograms. In: *2015 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 31–38. IEEE (2015)
13. Borkin, M., Gajos, K., Peters, A., Mitsouras, D., Melchionna, S., Rybicki, F., Feldman, C., Pfister, H.: Evaluation of artery visualizations for heart disease diagnosis. *IEEE Trans. Vis. Comput. Graph.* **17**(12), 2479–2488 (2011)
14. Borkin, M., Vo, A., Bylinskii, Z., Isola, P., Sunkavalli, S., Oliva, A., Pfister, H., et al.: What makes a visualization memorable? *IEEE Trans. Vis. Comput. Graph.* **19**(12), 2306–2315 (2013)
15. Chapman, P., Stapleton, G., Rodgers, P., Micallef, L., Blake, A.: Visualizing sets: an empirical comparison of diagram types. In: Dwyer, T., Purchase, H., Delaney, A. (eds.) *Diagrams 2014*. LNCS (LNAI), vol. 8578, pp. 146–160. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44043-8_18
16. Christensen, J., Bae, J.H., Watson, B., Rappa, M.: Understanding which graph depictions are best for viewers. In: Christie, M., Li, T.-Y. (eds.) *SG 2014*. LNCS, vol. 8698, pp. 174–177. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11650-1_17
17. Dinkla, K., Westenberg, M.A., van Wijk, J.J.: Compressed adjacency matrices: untangling gene regulatory networks. *IEEE Trans. Vis. Comput. Graph.* **18**(12), 2457–2466 (2012)
18. Ellson, J., Gansner, E., Koutsofios, L., North, S.C., Woodhull, G.: Graphviz—open source graph drawing tools. In: Mutzel, P., Jünger, M., Leipert, S. (eds.) *GD 2001*. LNCS, vol. 2265, pp. 483–484. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45848-4_57
19. Elmqvist, N., Do, T.N., Goodell, H., Henry, N., Fekete, J.D.: Zame: interactive large-scale graph visualization. In: *2008 IEEE Pacific Visualization Symposium, PacificVIS 2008*, pp. 215–222. IEEE (2008)
20. Fekete, J.D.: Reorder.js: a javascript library to reorder tables and networks. In: *IEEE VIS 2015* (2015)
21. Ghoniem, M., Fekete, J.D., Castagliola, P.: A comparison of the readability of graphs using node-link and matrix-based representations. In: *2004 IEEE Symposium on Information Visualization, INFOVIS 2004*, pp. 17–24. IEEE (2004)
22. Ghoniem, M., Fekete, J.D., Castagliola, P.: 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)
23. Heer, J., Bostock, M.: Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 203–212. ACM (2010)
24. Henry, N., Fekete, J.D.: Matrixexplorer: a dual-representation system to explore social networks. *IEEE Trans. Vis. Comput. Graph.* **12**(5), 677–684 (2006)
25. Hu, Y., Gansner, E., Kobourov, S.G.: Visualizing graphs and clusters as maps. *IEEE Comput. Graph. Appl.* **30**(6), 54–66 (2010)

26. Huang, W.: Using eye tracking to investigate graph layout effects. In: 2007 6th International Asia-Pacific Symposium on Visualization, APVIS 2007, pp. 97–100. IEEE (2007)
27. Huang, W., Eades, P., Hong, S.H.: Measuring effectiveness of graph visualizations: a cognitive load perspective. *Inf. Vis.* **8**(3), 139–152 (2009)
28. Jianu, R., Rusu, A., Hu, Y., Taggart, D.: How to display group information on node-link diagrams: an evaluation. *IEEE Trans. Vis. Comput. Graph.* **20**(11), 1530–1541 (2014)
29. Jourdan, F., Melançon, G.: Tool for metabolic and regulatory pathways visual analysis. In: *Electronic Imaging 2003*, pp. 46–55. International Society for Optics and Photonics (2003)
30. Keller, R., Eckert, C.M., Clarkson, P.J.: Matrices or node-link diagrams: which visual representation is better for visualising connectivity models? *Inf. Vis.* **5**(1), 62–76 (2006)
31. Kosara, R., Ziemkiewicz, C.: Do mechanical turks dream of square pie charts? In: *Proceedings of the 3rd BELIV 2010 Workshop: Beyond Time and Errors: Novel Evaluation Methods for Information Visualization*, pp. 63–70. ACM (2010)
32. Lee, B., Plaisant, C., Parr, C.S., Fekete, J.D., Henry, N.: Task taxonomy for graph visualization. In: *Proceedings of the 2006 AVI Workshop on Beyond Time and Errors: Novel Evaluation Methods for Information Visualization*, pp. 1–5. ACM (2006)
33. Mason, W., Suri, S.: Conducting behavioral research on Amazons mechanical turk. *Behav. Res. Methods* **44**(1), 1–23 (2012)
34. Melançon, G.: Just how dense are dense graphs in the real world?: a methodological note. In: *Proceedings of the 2006 AVI Workshop on Beyond Time and Errors: Novel Evaluation Methods for Information Visualization*, pp. 1–7. ACM (2006)
35. Micallef, L., Dragicevic, P., Fekete, J.D.: Assessing the effect of visualizations on bayesian reasoning through crowdsourcing. *IEEE Trans. Vis. Comput. Graph.* **18**(12), 2536–2545 (2012)
36. Okoe, M., Jianu, R.: Ecological validity in quantitative user studies—a case study in graph evaluation (2015)
37. Okoe, M., Jianu, R.: Graphunit: evaluating interactive graph visualizations using crowdsourcing. In: *Computer Graphics Forum*, vol. 34, pp. 451–460. Wiley Online Library (2015)
38. Paolacci, G., Chandler, J., Ipeirotis, P.G.: Running experiments on Amazon mechanical turk. *Judgment Decis. Making* **5**(5), 411–419 (2010)
39. Perin, C., Dragicevic, P., Fekete, J.D.: Revisiting bertin matrices: new interactions for crafting tabular visualizations. *IEEE Trans. Vis. Comput. Graph.* **20**(12), 2082–2091 (2014)
40. Purchase, H.: Which aesthetic has the greatest effect on human understanding? In: *Graph Drawing*. pp. 248–261. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63938-1_67
41. Purchase, H.C., Cohen, R.F., James, M.: Validating graph drawing aesthetics. In: Brandenburg, F.J. (ed.) *GD 1995*. LNCS, vol. 1027, pp. 435–446. Springer, Heidelberg (1996). <https://doi.org/10.1007/BFb0021827>
42. Robertson, G., Fernandez, R., Fisher, D., Lee, B., Stasko, J.: Effectiveness of animation in trend visualization. *IEEE Trans. Vis. Comput. Graph.* **14**(6) (2008)
43. Rodgers, P., Stapleton, G., Chapman, P.: Visualizing sets with linear diagrams. *ACM Trans. Comput.-Hum. Interact. (TOCHI)* **22**(6), 27 (2015)

44. Ross, J., Irani, L., Silberman, M., Zaldivar, A., Tomlinson, B.: Who are the crowdworkers?: shifting demographics in mechanical turk. In: CHI 2010 Extended Abstracts on Human Factors in Computing Systems, pp. 2863–2872. ACM (2010)
45. Rufiange, S., McGuffin, M.J., Fuhrman, C.P.: Treematrix: a hybrid visualization of compound graphs. In: Computer Graphics Forum, vol. 31, pp. 89–101. Wiley Online Library (2012)
46. Saket, B., Scheidegger, C., Kobourov, S., Börner, K.: Map-based visualizations increase recall accuracy of data. *Comput. Graph. Forum* **34**(3), 441–450 (2015)
47. Saket, B., Simonetto, P., Kobourov, S.: Group-level graph visualization taxonomy. arXiv preprint [arXiv:1403.7421](https://arxiv.org/abs/1403.7421) (2014)
48. Saket, B., Simonetto, P., Kobourov, S., Borner, K.: Node, node-link, and node-link-group diagrams: an evaluation. *IEEE Trans. Vis. Comput. Graph.* **20**(12), 2231–2240 (2014)
49. Sedlmair, M., Isenberg, P., Baur, D., Mauerer, M., Pigorsch, C., Butz, A.: Cardiomgram: visual analytics for automotive engineers. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 1727–1736. ACM (2011)
50. Shannon, P., Markiel, A., Ozier, O., Baliga, N.S., Wang, J.T., Ramage, D., Amin, N., Schwikowski, B., Ideker, T.: Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.* **13**(11), 2498–2504 (2003)
51. Shen, Z., Maz, K.L.: Path visualization for adjacency matrices. In: Proceedings of the 9th Joint Eurographics/IEEE VGTC conference on Visualization, pp. 83–90. Eurographics Association (2007)
52. Shneiderman, B.: The eyes have it: a task by data type taxonomy for information visualizations. In: 1996 Proceedings of the IEEE Symposium on Visual Languages, pp. 336–343. IEEE (1996)
53. Viégas, F.B., Donath, J.: Social network visualization: can we go beyond the graph. In: Workshop on Social Networks, CSCW, vol. 4, pp. 6–10 (2004)
54. Von Landesberger, T., Kuijper, A., Schreck, T., Kohlhammer, J., van Wijk, J.J., Fekete, J.D., Fellner, D.W.: Visual analysis of large graphs: state-of-the-art and future research challenges. In: Computer Graphics Forum, vol. 30, pp. 1719–1749. Wiley Online Library (2011)
55. Ware, C., Purchase, H., Colpoys, L., McGill, M.: Cognitive measurements of graph aesthetics. *Inf. Vis.* **1**(2), 103–110 (2002)
56. Yi, J.S., Elmqvist, N., Lee, S.: Timematrix: analyzing temporal social networks using interactive matrix-based visualizations. *Int. J. Hum.-Comput. Interact.* **26**(11–12), 1031–1051 (2010)
57. Ziemkiewicz, C., Kosara, R.: The shaping of information by visual metaphors. *IEEE Trans. Vis. Comput. Graph.* **14**(6) (2008)

Tree Drawings

Improved Bounds for Drawing Trees on Fixed Points with L-Shaped Edges

Therese Biedl¹(✉), Timothy M. Chan², Martin Derka¹, Kshitij Jain¹,
and Anna Lubiw¹

¹ University of Waterloo, Waterloo, ON N2L 3G1, Canada
{biedl,mderka,k22jain,alubiw}@uwaterloo.ca

² University of Illinois at Urbana-Champaign, Urbana, IL, USA
tmc@illinois.edu

Abstract. Let T be an n -node tree of maximum degree 4, and let P be a set of n points in the plane with no two points on the same horizontal or vertical line. It is an open question whether T always has a planar drawing on P such that each edge is drawn as an orthogonal path with one bend (an “L-shaped” edge). By giving new methods for drawing trees, we improve the bounds on the size of the point set P for which such drawings are possible to: $O(n^{1.55})$ for maximum degree 4 trees; $O(n^{1.22})$ for maximum degree 3 (binary) trees; and $O(n^{1.142})$ for perfect binary trees.

Drawing ordered trees with L-shaped edges is harder—we give an example that cannot be done and a bound of $O(n \log n)$ points for L-shaped drawings of ordered caterpillars, which contrasts with the known linear bound for unordered caterpillars.

1 Introduction

The problem of drawing a planar graph so that its vertices are restricted to a specified set of points in the plane has been well-studied, both from the perspective of algorithms and from the perspective of bounding the size of the point set and/or the number of bends needed to draw the edges. Throughout this paper we consider the version of the problem where the points are unlabelled, i.e., we may choose to place any vertex at any point.

One basic result is that every planar n -vertex graph has a planar drawing on any set of n points, even with the limitation of at most 2 bends per edge [11]. If every edge must be drawn as a straight-line segment then any n points in general position still suffice for drawing trees [4] and outerplanar graphs [3] but this result does not extend to any non-outerplanar graph [9], and the decision version of the problem becomes NP-complete [5]. Since n points do not always suffice, the next natural question is: How large must a *universal* point set be, and how many points are needed for *any* point set to be universal? An upper bound of $O(n^2)$ follows from the 1990 algorithms that draw planar graphs on an $O(n) \times O(n)$ grid [8, 13], but the best known lower bound, dating from 1989, is $c \cdot n$ for some $c > 1$ [6].

Although orthogonal graph drawing has been studied for a long time, analogous questions of universal point sets for orthogonal drawings have only been explored recently, beginning with Katz et al. [10] in 2010. Since at most 4 edges can be incident to a vertex in an orthogonal drawing, attention is restricted to graphs of maximum degree 4. Throughout the paper we will restrict attention to point sets in “general orthogonal position” meaning that no two points share the same x - or y -coordinate. We study the simplest type of orthogonal drawings where every edge must be drawn as an orthogonal path of two segments. Such a path is called an “L-shaped edge” and these drawings are called “planar L-shaped drawings”. Observe that any planar L-shaped drawing lives in the grid formed by the n horizontal and n vertical lines through the points.

Di Giacomo et al. [7] introduced the problem of planar L-shaped drawings and showed that any tree of maximum degree 4 has a planar L-shaped drawing on any set of $n^2 - 2n + 2$ points (in general orthogonal position, as will be assumed henceforth). Aichholzer et al. [1] improved the bound to $O(n^c)$ with $c = \log_2 3 \approx 1.585$. It is an open question whether n points always suffice. Surprisingly, nothing better is known for trees of maximum degree 3.

The largest subclass of trees for which n points are known to suffice is the class of caterpillars of maximum degree 3 [7]. A *caterpillar* is a tree such that deleting the leaves gives a path, called the *spine*. For caterpillars of maximum degree 4 with n nodes, any point set of size $3n - 2$ permits a planar L-shaped drawing [7], and the factor was improved to $5/3$ by Scheucher [12].

1.1 Our Results

We give improved bounds as shown in Table 1. A tree of max degree 3 (or 4) is *perfect* if it is a rooted binary tree (or ternary tree, respectively) in which all leaves are at the same height and all non-leaf nodes have 2 (or 3, respectively) children. Our bounds are achieved by constructing the drawings recursively and analyzing the resulting recurrence relations, which is the same approach used previously by Aichholzer et al. [1]. Our improvements come from more elaborate drawing methods. Results on maximum degree 3 trees are in Sect. 3 and results on maximum degree 4 trees are in Sect. 4.

Table 1. Previous and new bounds on the number of points sufficient for planar L-shaped drawings of any tree of n nodes. The previous bounds all come from Aichholzer et al. [1].

	Previous	New
deg 3 perfect	$n^{1.585}$	$n^{1.142}$
deg 3 general	$n^{1.585}$	$n^{1.22}$
deg 4 perfect	$n^{1.465}$ ^a	
deg 4 general	$n^{1.585}$	$n^{1.55}$

^aThe bound of $n^{1.465}$ is not explicit in [1] but will be explained in Sect. 4.

We also consider the case of *ordered* trees where the cyclic order of edges incident to each vertex is specified. We give an example of an n -node ordered tree (in fact, a caterpillar) and a set of n points such that the tree has no L-shaped planar drawing on the point set. We also give a positive result about drawing some ordered caterpillars on $O(n \log n)$ points. The caterpillars that can be drawn on such $O(n \log n)$ points include our example that cannot be drawn on a given set of n points. These results are in Sect. 2.

1.2 Further Background

Katz et al. [10] introduced the problem of drawing a planar graph on a specified set of points in the plane so that each edge is an *orthogeodesic path*, i.e. a path of horizontal and vertical segments whose length is equal to the L_1 distance between the endpoints of the path. They showed that the problem of deciding whether an n -vertex planar graph has a planar orthogeodesic drawing on a given set of n points is NP-complete. Subsequently, Di Giacomo et al. [7] showed that any n -node tree of maximum degree 4 has an orthogeodesic drawing on any set of n points where the drawing is restricted to the $2n \times 2n$ grid that consists of the “basic” horizontal and vertical lines through the points, plus one extra line between each two consecutive parallel basic lines. If the drawing is restricted to the basic grid, their bounds were $4n - 3$ points for degree-4 trees, and $3n/2$ points for degree-3 trees. These bounds were improved by Scheucher [12] and then by Bárány et al. [2].

2 Ordered Trees—The Case of Caterpillars

All previous work has assumed that trees are unordered, i.e., that we may freely choose the cyclic order of edges incident to a vertex. In this section we show that ordered trees on n vertices do not always have planar L-shaped drawings on n points. Our counterexample is a *top-view caterpillar*, i.e., a caterpillar such that the two leaves adjacent to each vertex lie on opposite sides of the spine. The main result in this section is that every top-view caterpillar has a planar L-shaped drawing on $cn \log n$ points for some $c > 0$.

First the counterexample. We prove the following in the full version:

Lemma 1. *The top-view caterpillar C_{14} on $n = 14$ nodes shown in Fig. 1(a) cannot be drawn with L-shaped edges on the point set P_{14} of size 14 shown in Fig. 1(c).*

It is conceivable that this counter-example is an isolated one—we have been unable to extend it to a family of such examples.

Next we explore the question of how many points are needed for a planar L-shaped drawing of an n -vertex top-view caterpillar. Consider the appearance of the caterpillar’s spine (a path) in such a drawing. Each node of the spine, except for the two endpoints, must have its two incident spine edges aligned—both horizontal or both vertical. Define a *straight-through* drawing of a path to

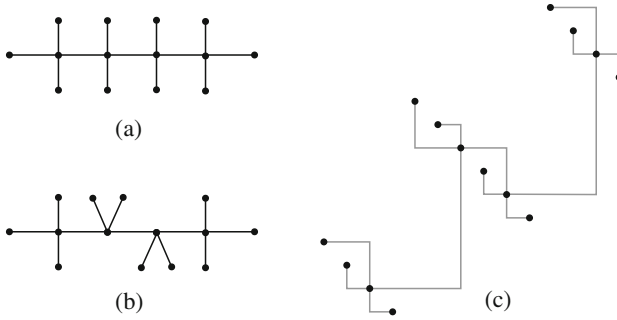


Fig. 1. The ordered top-view caterpillar C_{14} shown in (a) does not have a planar L-shaped drawing on the point set P_{14} shown in (c). The ordering shown in (b) does.

be a planar L-shaped drawing such that the two incident edges at each vertex are aligned. The best bound we have for the number of points that suffice for a straight-through drawing of a path is obtained when we draw the path in a monotone fashion, i.e. with non-decreasing x -coordinates.

Theorem 1. *Any path of n vertices has an x -monotone straight-through drawing on any set of at least $c \cdot n \log n$ points for some constant c .*

Proof. We prove that if the number of points satisfies the recurrence $M(n) = 2M(\frac{n}{2}) + 2n$ then any path of n vertices has an x -monotone straight-through drawing on the points. Observe that this recurrence relation solves to $M(n) \in \Theta(n \log n)$ which will complete the proof. Within a constant factor we can assume without loss of generality that n is a power of 2.

Order the points by x -coordinate. Recall our assumption that no two points share the same x - or y -coordinate. By induction, the first half of the path has an x -monotone straight-through drawing on the first $M(\frac{n}{2})$ points. We add the assumption that the path starts with a horizontal segment.

Let p be the second last point used. Since n is a power of 2, the path goes through p on a horizontal segment. Let T be the set of points to the right of and above p . Let B be the set of points to the right of and below p . Refer to Fig. 2(a). In T , consider the partial order $(x_1, y_1) \prec_T (x_2, y_2)$ if $x_1 < x_2$ and $y_1 < y_2$. Let T' be the set of minimal elements in this partial order. Similarly, in B , let B' be the set of elements that are minimal in the ordering $(x_1, y_1) \prec_B (x_2, y_2)$ if $x_1 < x_2$ and $y_1 > y_2$. If T' has n or more points, then we can draw the whole path on T' with an x -monotone straight-through drawing starting with a horizontal segment. The same holds if B' has n or more points. Thus we may assume that $|T'|, |B'| < n$. We now remove T' and B' ; let $R = (T - T') \cup (B - B')$. Then $|R| \geq M(\frac{n}{2})$.

By induction the second half of the path has an x -monotone straight-through drawing on the set R starting with a horizontal segment. Let r be the first point used for this drawing. Assume without loss of generality that r lies in T . (The other case is symmetric.) Consider the rectangle with opposite corners at p and

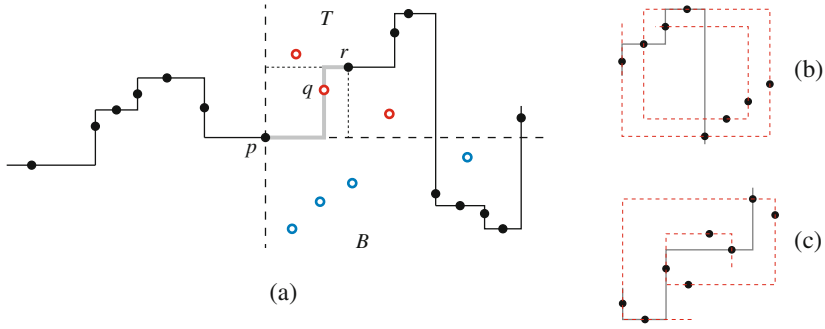


Fig. 2. (a) The construction for the proof of Theorem 1. The points of T' and B' are drawn as hollow red points above p and hollow blue points below p , respectively. (b-c) Examples of point sets of size $2n$ for which the maximum length of an x -monotone straight-through path is $n + 1$. Such paths are shown in grey. In both cases there are non-monotone planar straight-through paths of length $2n$ (dashed). (Color figure online)

r . Since r is not in T' , there is a point $q \in T$ inside the rectangle. We can join the two half paths using a vertical segment through q and the last vertex of the first half path is embedded at q . \square

We can extend the above result to draw the entire caterpillar (not just its spine) with the same bound on the number of points:

Theorem 2. *Any top-view caterpillar of n vertices has a planar L-shaped drawing on any set of $c \cdot n \log n$ points for some constant c .*

Proof (outline). Follow the above construction, but in addition to T' and B' , also take the second and third layers. If any layer has n or more points, we embed the whole caterpillar on it [7]. Otherwise, we remove at most a linear number of points, and embed the second half of the caterpillar by induction on the remaining points. Then, in the rectangle between p and r there must be an increasing sequence of 3 points. Use the middle one for the left-over spine-vertex q and the other two for the leaves of q . \square

We conjecture that $2n$ points suffice for an x -monotone straight-through drawing of any n -path. See Fig. 2(b-c) for a lower bound of $2n$. Do n points suffice if the x -monotone condition is relaxed to planarity? Finally, we mention that the problem of finding monotone straight-through paths is related to a problem about alternating runs in a sequence, as explained in the full paper.

3 Trees of Maximum Degree 3

In this section, we prove bounds on the number of points needed for L-shaped drawings of trees with maximum degree 3. We treat the trees as rooted and thus,

we refer to them as binary trees. We name the parts of the tree as shown in Fig. 3(a). The root r_0 has two subtrees T_1 and T_2 of size n_1 and n_2 , respectively, with $n_1 \leq n_2$. T_2 's root, r_1 , has subtrees of sizes $n_{2,1}$ and $n_{2,2}$ with $n_{2,1} \leq n_{2,2}$.

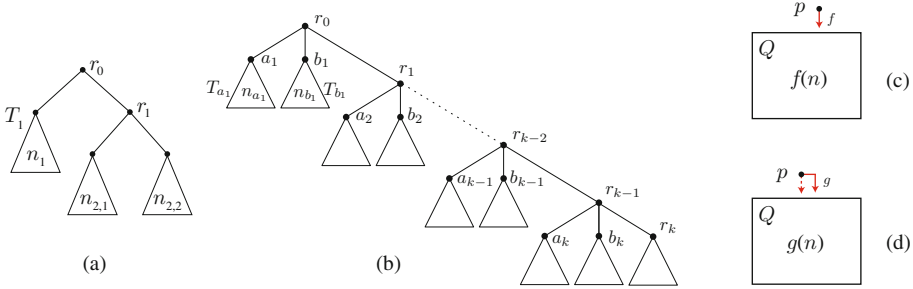


Fig. 3. The naming conventions for (a) binary and (b) ternary trees. The set-up for (c) f -configurations and (d) g -configurations.

The main idea is to draw a tree T on a set of points in a rectangle Q by partitioning the rectangle into subrectangles in which we recursively draw subtrees. This gives rise to recurrence relations for the number of points needed to draw trees of size n , which we then analyze. We distinguish two subproblems or “configurations.” In each, we must draw a tree T rooted at r_0 in a rectangle Q that currently has no part of the drawing inside it. Furthermore, the parent p of r_0 has already been drawn, and one or two rays outgoing from p have been reserved for drawing the first segment of edge (p, r_0) (without hitting any previous part of the drawing).

In the f -configuration the reserved ray from p goes vertically downward to Q . See Fig. 3(c). Let $f(n)$ be the smallest number of points such that any binary tree with n vertices can be drawn in any rectangle with $f(n) - 1$ points in the f -configuration¹. We will give various ways of drawing trees in the f -configuration, each of which gives an upper bound on $f(n)$. Among these choices, the algorithm uses the one that requires the fewest points.

In the g -configuration we reserve a horizontal ray from p , that allows the L-shaped edge (p, r_0) to turn downward into Q at any point without hitting any previous part of the drawing. In addition, we reserve the vertical ray downward from p in case this ray enters Q . See Fig. 3(d) for the case where the horizontal ray goes to the right. Let $g(n)$ be the smallest number of points such that any binary tree with n vertices can be drawn in any rectangle with $g(n) - 1$ points in the g -configuration. Observe that $f(n) \geq g(n)$ since the g -configuration gives us strictly more freedom.

We start with two easy constructions to give the flavour of our methods.

f -draw-1. This method, illustrated in Fig. 4(a), applies to an f -configuration. We first describe the construction and then say how many points are required.

¹ Beware: we will use the same notation $f(n)$ in Sect. 4 to refer to ternary trees.

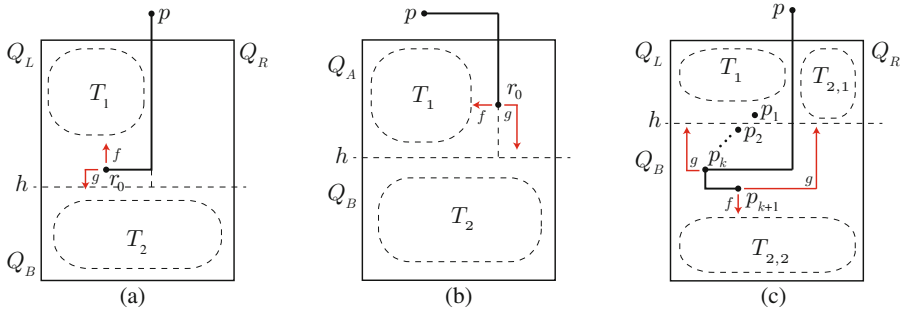


Fig. 4. Three methods: (a) f -draw-1; (b) g -draw; and (c) f -draw-2.

Continue the vertical ray from p downward to a horizontal half-grid line h determined as follows. Partition Q by h and the ray down to h into 3 parts: Q_B , the rectangle below h ; Q_L , the upper left rectangle; and Q_R , the upper right rectangle. Choose h to be the highest half-grid line such that Q_L or Q_R has $f(n_1)$ points. Without loss of generality, assume that Q_L has $f(n_1)$ points, and Q_R has at most $f(n_1)$ points. Place r_0 at the bottommost point of Q_L . Draw the edge (p, r_0) down and left. Start a ray vertically up from r_0 , and recursively draw T_1 in f -configuration (rotated 180°) in the subrectangle of Q_L above r_0 , which has $f(n_1) - 1$ points. This leaves the leftward and downward rays free at r_0 , so we can draw T_2 recursively in g -configuration in Q_B so long as there are $g(n_2) - 1$ points. The total number of points required is $2f(n_1) + g(n_2) - 1$. Recall that $f(n)$ is 1 more than the number of required points, so this proves:

$$f(n) \leq 2f(n_1) + g(n_2). \tag{f-1}$$

Observe that we could have swapped f and g which proves:

$$f(n) \leq 2g(n_1) + f(n_2). \tag{f-1'}$$

The above method can be viewed as a special case of Aichholzer et al.'s method for ternary trees [1] (see Sect. 4). We incorporate two new ideas to improve their result: first, they used only f -configurations, but we notice that one of the above two recursive subproblems is a g -configuration in the binary tree case, and can be solved by a better recursive algorithm; second, their method wasted all the points in Q_R , but we will give more involved constructions that allow us to use some of those points.

g -draw. This method applies to a g -configuration where the ray from the parent node p goes to the right. Partition Q at the highest horizontal half-grid line such that the top rectangle Q_A has $f(n_1)$ points. We separate into two cases depending whether the rightmost point, q , of Q_A is to the right or left of p .

If q is to the right of p , place r_0 at q , and draw the edge (p, r_0) right and down. See Fig. 4(b). Start a ray leftward from r_0 and recursively draw T_1 in f -configuration in the subrectangle of Q_A to the left of q . Note that there are $f(n_1) - 1$ points here, which is sufficient. The rightward and downward rays at r_0 are free, so we can draw T_2 recursively in g -configuration in Q_B if there are $g(n_2) - 1$ points. The total number of points required is $f(n_1) + g(n_2) - 1$.

If all points of Q_A lie to the left of p , then place r_0 at the bottommost point of Q_A and observe that we now have the situation of f -draw-1 with Q_R empty, and $f(n_1) + g(n_2) - 1$ points suffice.

This proves:

$$g(n) \leq f(n_1) + g(n_2). \tag{g}$$

We now describe a different f -drawing method that is more efficient than f -draw-1 above, and will be the key for our bound for binary trees.

f -draw-2. This method applies to an f -configuration. We begin as in f -draw-1, though with the f -drawing and the g -drawing switched. Partition Q by a horizontal half-grid line h and the ray from p down to h into 3 parts: Q_B , the rectangle below h ; Q_L , the upper left rectangle; and Q_R , the upper right rectangle. Choose h to be the highest half-grid line such that Q_L or Q_R has $g(n_1)$ points. Without loss of generality, assume the former. We separate into two cases depending on the size of Q_R .

If $|Q_R| < g(n_{2,1})$ then we follow the f -draw-1 method. Let p_1 be the bottommost point of Q_L . Place r_0 at p_1 , draw the edge (p, r_0) down and left, recursively draw T_1 in g -configuration in Q_L using leftward/upward rays from r_0 , and recursively draw T_2 in f -configuration in Q_B using a downward ray from r_0 . This requires $g(n_1) + g(n_{2,1}) + f(n_2) - 1$ points, where $g(n_{2,1})$ accounts for the wasted points in Q_R .

If $|Q_R| \geq g(n_{2,1})$ then we make use of the points in Q_R by drawing subtree $T_{2,1}$ there. Let p_1 be the bottommost point of Q_L , and let p_2, p_3, \dots be the points of Q_B below p_1 in decreasing y -order. Let $k \geq 2$ be the smallest index such that either $k = n$ or point p_{k+1} lies to the right of p_k . See Fig. 4(c).

We have two subcases. If $k = n$, then p_1, \dots, p_k form a monotone chain of length n , i.e., a diagonal point set in the terminology of Di Giacomo et al. [7]. They showed that any tree of n points can be embedded on a diagonal point set, so we simply draw T on these n points. (Note that if this construction is used in the induction step, upward visibility is needed for connecting T to the rest of the tree, and this can be achieved.)

Otherwise $k < n$. Place r_0 at point p_k and r_1 at p_{k+1} . Draw the edge (p, r_0) down and left, and the edge (r_0, r_1) down and right. Recursively draw T_1 in g -configuration in Q_L using leftward/upward rays from r_0 . Draw $T_{2,2}$ in f -configuration in the rectangle below r_1 using a downward ray from r_1 . Draw $T_{2,1}$ in g -configuration in Q_R using the rightward ray from r_1 . Observe that if r_1 lies to the right of p (i.e., below Q_R rather than below Q_L) then the upward ray

from r_1 is clear (as required for a g -drawing). The number of points required is at most $2g(n_1) + n + f(n_{2,2}) - 1$. This accounts for at most $g(n_1)$ points in Q_R , and at most n points below h and above r_1 .

This proves:

$$f(n) \leq \max\{g(n_1) + g(n_{2,1}) + f(n_2), 2g(n_1) + f(n_{2,2}) + n\}. \tag{f-2}$$

Theorem 3. *Any perfect binary tree with n nodes has an L-shaped drawing on any point set of size $c \cdot n^{1.142}$ for some constant c .*

Proof. For perfect binary trees we have $n_1 = n_2 = \frac{1}{2}n$ and $n_{2,1} = n_{2,2} = \frac{1}{4}n$. We solve the simultaneous recurrence relations for f and g in the full paper. \square

Theorem 4. *Any binary tree has an L-shaped drawing on any point set of size $c \cdot n^{1.22}$ for some constant c .*

Proof. For $n_1 \leq 0.349n$, we use recursion (f-1). For $n_{2,1} \leq 0.082n$, we combine recursion (f-1') and (f-1) to obtain $f(n) \leq 2g(n_1) + 2f(n_{2,1}) + g(n_{2,2})$. For $n_1 > 0.349n$ and $n_{2,1} > 0.082n$, we use recursion (f-2). We solve the simultaneous recurrence relations for f and g in the full paper. \square

4 Trees of Maximum Degree 4

In this section, we prove bounds on the number of points needed for L-shaped drawings of trees with maximum degree 4. We treat the trees as rooted and refer to them as ternary trees. Given a ternary tree of n nodes, let a_1, b_1 and r_1 be the three children of the root r_0 . We use T_v to denote the subtree rooted at a node v , and n_v to denote the number of nodes in T_v . We name the children of the root such that $n_{a_1} \leq n_{b_1} \leq n_{r_1}$. For $i \geq 2$, let a_i, b_i, r_i be the three children of r_{i-1} , named such that $n_{a_i} \leq n_{b_i} \leq n_{r_i}$. See Fig. 3(b).

We will draw ternary trees using only the f -configuration as defined in Sect. 3 (see Fig. 3(c)). In this section (as opposed to the previous one) we define $f(n)$ to be minimum number such that any ternary tree of n nodes can be drawn in f -configuration on any set of $f(n) - 1$ points.

As in Sect. 3, we will give various drawing methods, each of which gives a recurrence relation for $f(n)$. We begin with a re-description of Aichholzer et al.'s method [1].

f_4 -draw-1. Extend the vertical ray from p downward to a horizontal half-grid line h determined as follows. Partition Q by h and the ray down to h into 3 parts: Q_B , the rectangle below h ; Q_L , the upper left rectangle; and Q_R , the upper right rectangle. Choose h to be the highest half-grid line such that Q_L or Q_R has $2f(n_{a_1}) + 2f(n_{b_1})$ points. Without loss of generality, assume the former. Partition Q_L vertically into two rectangles Q_{LL} and Q_{LR} with at least $f(n_{a_1})$ points on the left and at least $f(n_{b_1})$ points on the right respectively, with Q_{LL} to the left of Q_{LR} . Place r_0 at the bottommost point in Q_{LR} . Extend a ray upward

from r_0 and recursively draw T_{b_1} on the remaining $f(n_{b_1}) - 1$ points in Q_{LR} . Extend a ray to the left from r_0 and recursively draw T_{a_1} on the $f(n_{a_1})$ points in Q_{LL} . Finally, extend a ray downward from r_0 and recursively draw T_{r_1} in Q_B . See Fig. 5(a). The number of points required is $2f(n_{a_1}) + 2f(n_{b_1}) + f(n_{r_1}) - 1$, so this proves:

$$f(n) \leq 2f(n_{a_1}) + 2f(n_{b_1}) + f(n_{r_1}). \tag{f_4-1}$$

For example, in the case when T is perfect (with $n_{a_1} = n_{b_1} = n_{r_1} = \frac{n}{3}$), the inequality (f₄₋₁) becomes $f(n) \leq 5f(n/3)$, which resolves to $O(n^{\log_3 5})$ and $\log_3 5 \approx 1.465$. The critical case for this recursion, though, turns out to be when $n_{a_1} = 0$ and $n_{b_1} = n_{r_1} = \frac{n}{2}$, which gives $f(n) \leq 3f(n/2)$ and leads to Aichholzer et al.’s $O(n^{\log_3 2})$ result.

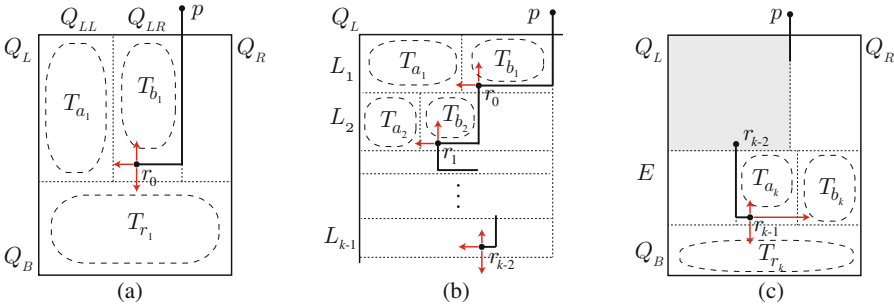


Fig. 5. (a) f_4 -draw-1. (b) Drawing the “small” subtrees in Q_L . (c) f_4 -draw-2A.

f_4 -draw-2. To improve their result, our idea again is to avoid wasting the points in Q_R , and use some of those points in the recursive drawings of subtrees at the next levels. However, simply considering subtrees at the second level is not sufficient for an asymptotic improvement if T_{a_2} and T_{b_2} are too small. Thus, we consider a more complicated approach that stops at the first level $k \geq 2$ where $n_{r_k} \leq 0.9n_{r_{k-1}}$. Note that for $i = 2, \dots, k - 1$, we have $n_{r_i} > 0.9n_{r_{i-1}}$ and $n_{a_i}, n_{b_i} \leq 0.1n_{r_{i-1}}$, and so T_{a_i} and T_{b_i} are “small” subtrees. We apply the same idea as above to draw not just T_{a_1}, T_{b_1} but also all the small subtrees T_{a_i} and $T_{b_i}, i = 2, \dots, k - 1$ in Q_L (appropriately defined), and then consider a few cases for how to draw the remaining “big” subtrees T_{a_k}, T_{b_k} , and T_{r_k} , possibly using some points in Q_R . The number of points we will need to reserve for drawing $T_{a_1}, T_{b_1}, \dots, T_{a_{k-1}}, T_{b_{k-1}}$ is

$$Y = f(n_{a_1}) + f(n_{b_1}) + \sum_{i=2}^{k-1} (2f(n_{a_i}) + 2f(n_{b_i})).$$

Extend the vertical ray from p downward until Q_L or Q_R has Y points. Without loss of generality, assume the former.

Drawing the Small Subtrees. We draw nodes r_i and subtrees T_{a_i} and T_{b_i} , $i = 1, \dots, k - 1$ in Q_L as follows. Split Q_L horizontally into rectangles L_1, \dots, L_{k-1} . The plan is to draw r_i, T_{a_i} and T_{b_i} in L_i , in the same way that T_{a_1} and T_{b_1} were drawn in f_4 -draw-1. See Fig. 5(b). Level L_1 is special because the vertical ray from p is at the right boundary of L_1 . Thus, we require $f(n_{a_1}) + f(n_{b_1})$ points. For levels L_i , $i = 2, \dots, k - 1$ the vertical ray from r_{i-2} may enter L_i at any point, so we require $2f(n_{a_i}) + 2f(n_{b_i})$ points to follow the plan of f_4 -draw-1, and the L-shaped edge from r_{i-2} to r_{i-1} may turn left or right. The total number of points we need in all levels is Y , which is why we defined Y as we did.

Drawing the Final Three Subtrees. It remains to draw r_{k-1} and its three subtrees T_{a_k}, T_{b_k} , and T_{r_k} . We will draw T_{r_k} on the bottommost $f(n_{r_k}) - 1$ points of Q . Call this rectangle Q_B . Let E be the “equatorial zone” that lies between Q_L, Q_R above and Q_B below. See Fig. 5(c). If we are lucky, then not too many points are wasted in Q_R . Let $Z \leq Y$ be the number of points in Q_R .

Case A: $Z < f(n_{b_k})$. In this case we draw r_{k-1}, T_{a_k} and T_{b_k} in E as in f_4 -draw-1. See Fig. 5(c). For this, we need $2f(n_{a_k}) + 2f(n_{b_k})$ points in E . The total number of points required in this case is $Y + Z + 2f(n_{a_k}) + 2f(n_{b_k}) + f(n_{r_k}) - 1$, so this proves:

$$\begin{aligned}
 f(n) &\leq Y + Z + 2f(n_{a_k}) + 2f(n_{b_k}) + f(n_{r_k}) && (f_4\text{-}2A) \\
 &= f(n_{a_1}) + f(n_{b_1}) + \sum_{i=2}^{k-1} (2f(n_{a_i}) + 2f(n_{b_i})) + 2f(n_{a_k}) + 3f(n_{b_k}) + f(n_{r_k}).
 \end{aligned}$$

We must now deal with the unlucky case when $Z \geq f(n_{b_k})$. We will require $3f(n_{a_k}) + f(n_{b_k})$ points in E . We sum up the total number of points below, but first we describe how to complete the drawing in E . Partition E into three regions: E_L, E_M , and E_R , where E_L is the region to the left of r_{k-2} , E_R is the region to the right of p , and E_M is the region between them. See Fig. 6. Observe that either $|E_L| \geq f(n_{a_k}) + f(n_{b_k})$, or $|E_M| \geq f(n_{a_k})$, or $|E_R| > f(n_{a_k})$. We show how to draw r_{k-1}, T_{a_k} and T_{b_k} in each of these 3 cases.

Case B1: $|E_L| \geq f(n_{a_k}) + f(n_{b_k})$. In this case we draw r_{k-1}, T_{a_k} and T_{b_k} in E_L as in f_4 -draw-1. See Fig. 6(a). Since E_L is to the left of the ray down from r_{k-2} , we have sufficiently many points.

Case B2: $|E_M| \geq f(n_{a_k})$. In this case we place r_{k-1} at the lowest point of E_M , draw T_{a_k} above it in E_M , and T_{b_k} to its right in $Q_R \cup E_R$. See Fig. 6(b). Since $|Q_R| = Z \geq f(n_{b_k})$, we have enough points to do this.

Case B3: $|E_R| > f(n_{a_k})$. In this case we place r_{k-1} at the leftmost point of E_R , draw T_{a_k} to its right in E_R and T_{b_k} above it in Q_R . See Fig. 6(c). Again, there are sufficiently many points.

The total number of points required in each of these three cases is $Y + Z + 3f(n_{a_k}) + f(n_{b_k}) + f(n_{r_k}) - 1$, and $Y \leq Z$ which yields:

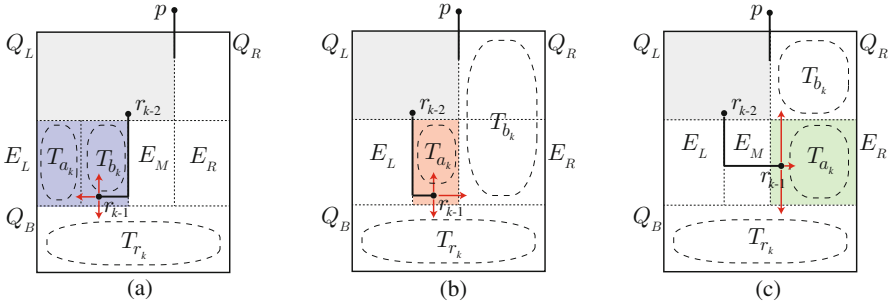


Fig. 6. The drawings for f_4 -draw-2B: (a) Case B1, with E_L in blue; (b) Case B2, with E_M in red; (c) Case B3, with E_R in green. (Color figure online)

$$\begin{aligned}
 f(n) &\leq Y + Z + 3f(n_{a_k}) + f(n_{b_k}) + f(n_{r_k}) && (f_4\text{-2B}) \\
 &= 2f(n_{a_1}) + 2f(n_{b_1}) + \sum_{i=2}^{k-1} (4f(n_{a_i}) + 4f(n_{b_i})) + 3f(n_{a_k}) + f(n_{b_k}) + f(n_{r_k}).
 \end{aligned}$$

The bound on $f(n)$ obtained from f_4 -draw-2 is the maximum of $(f_4\text{-2A})$ and $(f_4\text{-2B})$.

Theorem 5. Any ternary tree with n nodes has an L-shaped drawing on any point set of size $2n^{1.55}$.

Proof. For $n_{b_1} \leq 0.47n$, we use recursion $(f_4\text{-1})$. Otherwise, we use $(f_4\text{-2A})$ or $(f_4\text{-2B})$ and take the larger of the two bounds. We solve the recurrence relation for f in the full paper. \square

5 Conclusions

We have made slight improvements on the exponent t in the bounds that $c \cdot n^t$ points always suffice for drawing trees of maximum degree 4, or 3, with L-shaped edges. Improving the bounds to, e.g., $O(n \log n)$ will require a breakthrough. In the other direction, there is still no counterexample to the possibility that n points suffice.

We introduced the problem of drawing ordered trees with L-shaped edges, where many questions remain open. For example: Do $c \cdot n$ points suffice for drawing ordered caterpillars? Can our isolated example be expanded to prove that n points are not sufficient in general?

Acknowledgments. We thank Jeffrey Shallit for investigating the alternating sequences discussed in Sect. 2. This work was done as part of a Problem Session in the Algorithms and Complexity group at the University of Waterloo. We thank the other participants for helpful discussions.

References

1. Aichholzer, O., Hackl, T., Scheucher, M.: Planar L-shaped point set embeddings of trees. In: European Workshop on Computational Geometry (EuroCG) (2016). <http://www.eurocg2016.usi.ch/>
2. Bárány, I., Buchin, K., Hoffmann, M., Liebenau, A.: An improved bound for orthogeodesic point set embeddings of trees. In: European Workshop on Computational Geometry (EuroCG) (2016). <http://www.eurocg2016.usi.ch/>
3. Bose, P.: On embedding an outer-planar graph in a point set. *Comput. Geom.* **23**(3), 303–312 (2002)
4. Bose, P., McAllister, M., Snoeyink, J.: Optimal algorithms to embed trees in a point set. *J. Graph Algorithms Appl.* **1**(2), 1–15 (1997)
5. Cabello, S.: Planar embeddability of the vertices of a graph using a fixed point set is NP-hard. *J. Graph Algorithms Appl.* **10**(2), 353–363 (2006)
6. Chrobak, M., Karloff, H.: A lower bound on the size of universal sets for planar graphs. *ACM SIGACT News* **20**(4), 83–86 (1989)
7. Di Giacomo, E., Frati, F., Fulek, R., Grilli, L., Krug, M.: Orthogeodesic point-set embedding of trees. *Comput. Geom.* **46**(8), 929–944 (2013)
8. Fraysseix, H.D., Pach, J., Pollack, R.: How to draw a planar graph on a grid. *Combinatorica* **10**(1), 41–51 (1990)
9. Gritzmann, P., Mohar, B., Pach, J., Pollack, R.: Embedding a planar triangulation with vertices at specified points. *Am. Math. Monthly* **98**, 165–166 (1991)
10. Katz, B., Krug, M., Rutter, I., Wolff, A.: Manhattan-geodesic embedding of planar graphs. In: Eppstein, D., Gansner, E.R. (eds.) *GD 2009*. LNCS, vol. 5849, pp. 207–218. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11805-0_21
11. Kaufmann, M., Wiese, R.: Embedding vertices at points: few bends suffice for planar graphs. *J. Graph Algorithms Appl.* **6**(1), 115–129 (2002)
12. Scheucher, M.: Orthogeodesic point set embeddings of outerplanar graphs. Master’s thesis, Graz University of Technology (2015)
13. Schnyder, W.: Embedding planar graphs on the grid. In: *Proceedings of the First ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 138–148 (1990)

On Upward Drawings of Trees on a Given Grid

Therese Biedl¹(✉) and Debajyoti Mondal²

¹ Cheriton School of Computer Science, University of Waterloo,
Waterloo, ON, Canada
biedl@uwaterloo.ca

² Department of Computer Science, University of Saskatchewan,
Saskatoon, SK, Canada
dmondal@cs.usask.ca

Abstract. Computing a minimum-area planar straight-line drawing of a graph is known to be NP-hard for planar graphs, even when restricted to outerplanar graphs. However, the complexity question is open for trees. Only a few hardness results are known for straight-line drawings of trees under various restrictions such as edge length or slope constraints. On the other hand, there exist polynomial-time algorithms for computing minimum-width (resp., minimum-height) upward drawings of trees, where the height (resp., width) is unbounded.

In this paper we take a major step in understanding the complexity of the area minimization problem for strictly-upward drawings of trees, which is one of the most common styles for drawing rooted trees. We prove that given a rooted tree T and a $W \times H$ grid, it is NP-hard to decide whether T admits a strictly-upward (unordered) drawing in the given grid. The hardness result holds both in polyline and straight-line drawing settings.

1 Introduction

Drawing planar graphs on a small integer grid is an active research area [17], which is motivated by various practical needs such as for VLSI circuit layout and small-screen visualization. Trees are one of the most studied graph classes in this context. While computing a minimum-area planar straight-line drawing of an arbitrary planar graph is known to be NP-complete [11], even for planar graphs with bounded pathwidth [11] and outerplanar graphs [4], the problem seems very intriguing for trees.

In this paper we examine *rooted and unordered trees*, i.e., one of the vertices is designated as the root and the left to right order of the children can be chosen arbitrarily. A natural way to display such a tree is to compute a (*strictly upward*) drawing, where each vertex is mapped to an integer grid point such that the parents have (strictly) larger y -coordinates than their children, each edge is drawn with y -monotone polylines with bends at integer grid points, and no two edges cross except possibly at their common endpoint. The *width*, *height* and *area*

Work of the authors supported in part by NSERC.

of the drawing are respectively the width, height, and area of the smallest axis-parallel rectangle that encloses the drawing. In a *straight-line (strictly) upward drawing*, the edges are restricted to be straight line segments.

We refer the reader to [17, Chapt. 5] or [8] for a survey on small-area drawings of trees. Here we review only the results that focus on exact minimization. In the fixed-embedding setting, there exist polynomial-time algorithms to compute minimum-area drawings for certain classes of planar graphs [4, 15], namely, those that simultaneously have bounded treewidth and bounded face-degrees. On the other hand, the problem becomes NP-hard as soon as one of the above constraints is dropped [4]. The intractability of minimum-area tree drawings has been well established under some edge length and slope restrictions, e.g., when edges must be drawn with unit length or the slopes of the edges in the drawing must belong to one of the $(k/2)$ slopes determined by a k -grid. Note that an orthogonal grid is a 4-grid. Determining whether a tree can be drawn on a given k -grid, where $k \in \{4, 6, 8\}$ with edges of unit length is an NP-complete problem [2, 3, 10]. Similar hardness results hold also for ordered trees under various aesthetic requirements [7].

Not much is known about minimum-area upward drawings of trees. Trevisan [18] showed that every complete tree (resp., Fibonacci tree) with n vertices admit a strictly-upward straight-line drawing in $n + O(\log n \sqrt{n})$ (resp., $1.17n + O(\log n \sqrt{n})$) area, and conjectured that exact minimization may be possible in polynomial time. Trevisan mentioned that the problem of minimum-area strictly-upward drawing of a complete tree is a ‘sparse problem’. Therefore, proving this NP-hard would imply $P = NP$ by Mahaney’s theorem [12].

Interestingly, there exist polynomial-time algorithms for computing minimum-width upward drawings of trees, where the height is unbounded [5], and minimum-height drawings where the width is unbounded [1] (even for non-upward drawings [14]). Marriott and Stuckey [13] showed that minimizing the width of a strictly-upward straight-line drawing is NP-hard under the additional constraint that the x -coordinate of a parent is the average of the x -coordinates of its children. This additional requirement implies that a vertex with a single child must be placed directly above the child. Minimizing the width is known to be NP-hard even for ordered tree drawings under several other constraints [16].

We show that computing a strictly-upward drawing of a tree that resides in a given $W \times H$ -grid is NP-hard. Formally, we consider the following problem:

Problem: STRICTLY-UPWARD TREE DRAWING (SUTD)

Instance: An unordered rooted tree T , and two natural numbers W and H .

Question: Does T admit a strictly-upward (polyline or straight-line) drawing with width at most W and height at most H ?

2 NP-Hardness

We prove the NP-hardness of SUTD by a reduction from the following problem:

Problem: NUMERICAL 3-DIMENSIONAL MATCHING (N3DM)

Instance: Positive integers r_i, g_i, b_i , where $1 \leq i \leq k$, and an integer B such that $\sum_{i=1}^k (r_i + b_i + g_i) = k \cdot B$.

Question: Do there exist permutations π and π' of $\{1, \dots, k\}$ such that $r_{\pi(i)} + b_i + g_{\pi'(i)} = B$ for all $1 \leq i \leq k$?

N3DM is strongly NP-complete [9], and remains NP-complete even if we require all b_i 's to be odd, and the b_i 's are large and the g_i 's are huge relative to the r_i 's. (More precisely, $3k^c \leq r_i \leq 4k^c$, $k^{2c} \leq b_i \leq k^{2c} + k^c$, and $k^{4c} \leq g_i \leq k^{4c} + k^c$, where $c > 1$ is a constant.) See the full version [6] for further details.

Idea of the Reduction: One crucial ingredient is to construct a tree whose height matches the height-bound H of the drawing; this determines the layer for all vertices on paths to the deepest leaves because in a strictly-upward drawing every edge must lead to a layer farther down. Another crucial ingredient is to add so many vertices to the tree that (other than on the topmost layer) all grid-points must have a vertex on them. The next ingredient is to add “walls” that force the given $W \times H$ -grid to be divided into k regions that have B grid points available in each. (These walls are simply high-degree vertices, but since every grid-point must be used, nearly all neighbours of such a vertex must be in the layer below.) Finally we add gadgets that encode r_i, b_i, g_i in such a way that b_i vertices must be in the i th region defined by the walls, while r_i and g_i can freely choose into which of the regions they fall. Therefore, the division of them into the regions gives rise to a solution to N3DM.

Construction of T : Given an instance \mathcal{I} of N3DM, we construct a tree T with height $2k + 3$ as follows. We start with the *spinal path* v_1, \dots, v_{2k+3} and choose v_1 to be the root of T . We add two *supporting paths* u_2, \dots, u_{2k+3} and w_2, \dots, w_{2k+3} where u_2 and w_2 are children of v_1 . We set $H := 2k + 3$; hence in any strictly-upward drawing, u_i, v_i, w_i must be on layer ℓ_i for $2 \leq i \leq 2k + 3$. (We count the layers from top to bottom, i.e., layer ℓ_1 is the layer with y -coordinate H that contains the root v_1 , and ℓ_i is the layer whose y -coordinate is $H - i + 1$.)

Next we add the “walls” as shown in Fig. 1(a). Namely, add $B + 1$ leaves to T that are children of the root; the star graph induced by v_1 and its children is called the *wall of v_1* with *wall root* v_1 . The *wall children* of v_1 are these $B + 1$ added leaves, as well as child v_2 . Similarly we add a *wall of v_{2j}* for $j \in \{1, \dots, k + 1\}$, by adding $B + 1$ leaves to T that are children of v_{2j} so that v_{2j} has $B + 2$ wall children (including v_{2j+1}).

Finally, we encode r_i, g_i, b_i of the N3DM instance. For $1 \leq i \leq k$, we add $b_i - 1$ leaves to T and make them children of v_{2i+1} , see Fig. 1(b). We will call v_{2i+1} a *blue parent* and its b_i children (including v_{2i+2}) the *blue children* of v_{2i+1} . For each r_i (resp., g_i), we create a star with r_i (resp., g_i) leaves and connect the center

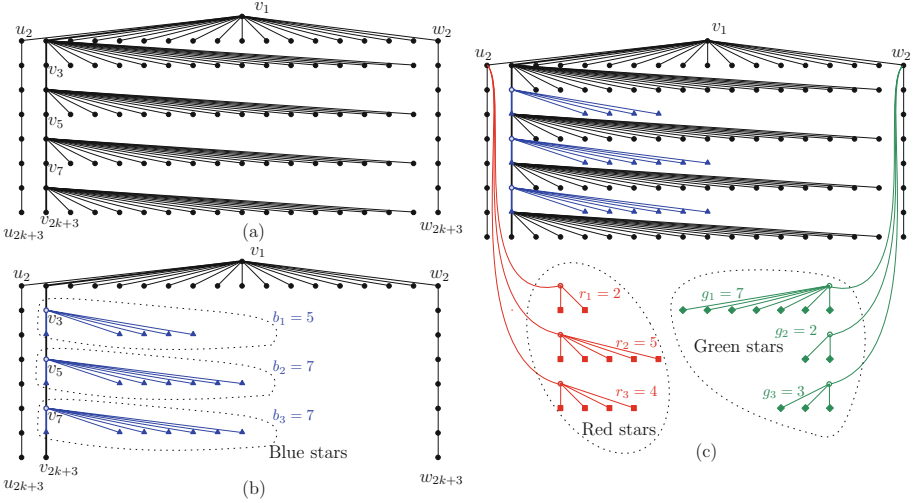


Fig. 1. (a) Illustration for the spinal path, supporting paths and walls. (b) Construction of the blue vertices. (c) Construction of the red and green vertices. For space reasons, the numbers in this example do not satisfy the constraints that we imposed in N3DM. (Color figure online)

to u_2 (resp., w_2), see Fig. 1(c). We refer to the stars corresponding to r_i and g_i as the *red* and *green stars*, respectively. This finishes the construction of tree T . Set $W := B + 4$ and observe that T has $2kB + 2B + 8k + 9 = (B + 4)(2k + 2) + 1 = W \times (H - 1) + 1$ vertices, which means that in any strictly-upward drawing in a $W \times H$ -grid, all layers except the top one must be completely full because the top layer can contain only one vertex (the root v_1).

From N3DM to Tree Drawing: If \mathcal{I} is a yes-instance, then we create a straight-line strictly-upward drawing of T on a $W \times H$ grid as illustrated in Fig. 2. The root v_1 is anywhere in the top layer. We place the supporting paths in layers $\ell_2, \dots, \ell_{2k+3}$ in the leftmost (resp., rightmost) column, as forced by the strictly-upward constraints. Vertex v_1 has $B + 4$ children (u_2, w_2 and the $B + 2$ wall children); we place all these in the second layer. We do not yet pick which of these $B + 2$ wall children becomes v_2 ; this will be determined later.

The solution to \mathcal{I} gives two permutations π, π' . Place the red, blue and green parents corresponding to $r_{\pi(i)}$, b_i and $g_{\pi'(i)}$ on layer ℓ_{2i+1} . More precisely, from left to right, we have first u_{2i} , then the red parent, then B wall children of v_{2i} , then the green parent, and then w_{2i} . Later, we will choose one of these B children to be v_{2i+1} , hence the blue parent corresponding to b_i . Observe that layer ℓ_{2i+1} is completely filled with vertices of T .

Place the red/blue/green children corresponding to $r_{\pi(i)}$, b_i and $g_{\pi'(i)}$ on layer ℓ_{2i+2} . More precisely, from left to right, we have first u_{2i} , then $r_{\pi(i)}$ red children, then one wall child of v_{2i} , then b_i blue children of v_{2i+1} , then another wall child of v_{2i} , then $g_{\pi'(i)}$ green children and finally w_{2i} . Since $r_{\pi(i)} + b_i + g_{\pi'(i)} = B$, layer ℓ_{2i+2} is also completely filled.

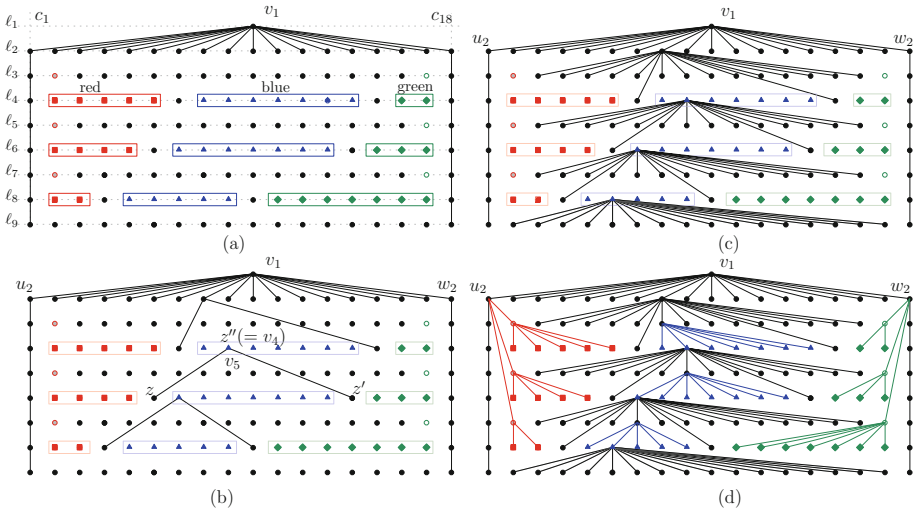


Fig. 2. Construction of a drawing of T on a $W \times H$ grid. (a) Placement of the vertices of T , drawing of the edges of the supporting paths, and the wall of v_1 . (b)–(c) Drawing of the blue stars and the remaining walls. (d) Drawing of the red and green stars. (Color figure online)

We must argue that all edges can be connected straight-line. This holds for the red stars since red children lie in the layer below their parent, and red parents lie in the column next to u_2 . See also Fig. 2(d). Similarly we can connect green stars. For the blue stars, because the b_i 's are much larger than the r_i 's, one can argue that the blue intervals of children of v_{2i-1} and v_{2i+1} overlap. We pick v_{2i} to be within this overlap in such a way that it can connect to the two wall children that are on layer ℓ_{2i+2} without using up grid points. We use as v_{2i+1} the point directly below v_{2i} ; due to the choice of v_{2i} then v_{2i+1} can connect to the blue interval on layer ℓ_{2i+2} without crossing. Details are in the full version [6].

From Tree Drawing to N3DM: We now show that any strictly-upward polyline drawing of T in a $W \times H$ -grid gives rise to permutations π and π' such that $r_{\pi(i)} + b_i + g_{\pi'(i)} = B$ holds for $1 \leq i \leq k$. We select $r_{\pi(i)}$ and $g_{\pi'(i)}$ in a bottom-up fashion, i.e., we first construct the triple $(r_{\pi(k)}, b_k, g_{\pi'(k)})$, then the triple $(r_{\pi(k-1)}, b_{k-1}, g_{\pi'(k-1)})$, and so on.

Since H equals the height of the tree, we know that v_i, u_i, w_i (for $i > 1$) are on layer ℓ_i , and the wall children of v_{2k+2} are on layer ℓ_{2k+3} (the bottommost layer). Hence layer ℓ_{2k+3} contains these $B + 2$ wall children, as well as u_{2k+3} and w_{2k+3} . By $W = B + 4$ this layer is full and contains no other vertices. Also v_{2k} lies on layer $2k$, and so all its wall children must be on layers ℓ_{2k+1} and ℓ_{2k+2} . We need an observation that crucially requires that all grid points are used. Details are in the full version [6].

Lemma 1. *Presume we know that all wall children of v_{2i} are on layers ℓ_{2i+1} and ℓ_{2i+2} . Then at most two wall children of v_{2i} are on layer ℓ_{2i+2} , and the wall children of v_{2i} on layer ℓ_{2i+1} occupy a consecutive set of points.*

Thus there are at least B wall children of v_{2k} that form an interval on ℓ_{2k+1} . Also u_{2k+1} and w_{2k+1} are on layer ℓ_{2k+1} , leaving at most two points in this layer free to be used for red or green stars, or blue or wall children from higher up.

We argue that indeed these two points must be used for a red parent and a green parent. To see this, consider the interval λ_m that consists of the middle $W-16=B-12$ points on layer ℓ_{2k+2} . T has $O(k^{2c+1})$ vertices that are not in green stars while $B > 3k^c + k^{2c} + k^{4c}$ so there must exist a green vertex in λ_m . It must be a green leaf l_g since layer ℓ_{2k+3} is full.

We claim that if a vertex x uses a point in λ_m , then its parent p_x is either v_{2k} or p_x is on layer ℓ_{2k+1} , i.e., one layer above x . To see this, assume otherwise and consider the place where edge (x, p_x) traverses layer ℓ_{2k+1} . If $p_x \neq v_{2k}$ then this point must be outside the interval of the $B = W - 4$ wall children of v_{2k} in this layer, else we would intersect an edge. So this point is within the leftmost or rightmost four units of ℓ_{2k+1} . Since λ_m covers all but the leftmost and rightmost 8 points of layer ℓ_{2k+2} , this forces p_x to be outside the allotted width of the drawing, a contradiction. See the details in the full version [6].

Consequently, the green parent p_g of l_g is on layer ℓ_{2k+1} and its green children are all on ℓ_{2k+2} . Due to our assumptions on N3DM, these green children, plus the blue children of v_{2k+1} , are not enough to fill λ_m , but leave too little space to place another green star. Therefore some point in λ_m must be occupied by a red child l_r , and its parent p_r hence is in layer ℓ_{2k+1} . Let $r_{\pi(k)}$ and $g_{\pi'(k)}$ be the numbers corresponding to the red and green stars at p_r and p_g .

We had $u_{2k+1}, w_{2k+1}, p_r, p_g$, and at least B wall children in ℓ_{2k+1} , which by $W = B + 4$ means that there are exactly B wall children in ℓ_{2k+1} and no other vertices. So two wall children are in layer ℓ_{2k+2} . These two, plus u_{2k+2} and w_{2k+2} , leave B points for the red, blue and green children, so $r_{\pi(k)} + b_k + g_{\pi'(k)} \leq B$. On the other hand, λ_m cannot contain any vertex x other than two wall-children of v_{2k} and these red, blue and green children, because layer ℓ_{2k+1} has no space left for the parent of x . So $r_{\pi(k)} + b_k + g_{\pi'(k)} \geq B - 14$. Since all input numbers are big enough, this implies $r_{\pi(k)} + b_k + g_{\pi'(k)} = B$, as desired.

It also follows that ℓ_{2k+1} is completely filled by these vertices, which means that no wall children of v_{2k-2} can be in layer ℓ_{2k+1} or below. We can now apply the same argument iteratively to compute the upper level triples $(r_{\pi(k-1)}, b_{k-1}, g_{\pi'(k-1)}), \dots, (r_{\pi(1)}, b_1, g_{\pi'(1)})$. This completes the NP-hardness reduction.

We can extract π and π' from any polyline drawing, while a solution to \mathcal{I} gives rise to a straight-line drawing. Therefore, the reduction holds both in polyline and straight-line drawing settings. Since SUTD is clearly in NP we hence have:

Theorem 1. *Given a tree T , and two natural numbers W and H , it is NP-complete to decide whether T admits a strictly-upward (polyline or straight-line) drawing with width at most W and height at most H .*

3 Directions for Future Research

Several interesting questions remain. In our reduction, we crucially used that the underlying tree has high degree, that the height of the drawing equals the height of the tree, that the order among children is not fixed, and that the drawing is *strictly* upward. Does SUTD remain NP-hard for bounded degree trees? What if the given width is optimal for the tree and the height needs to be minimized? How about order-preserving drawings and/or upward drawings?

The above questions are open also for many other popular styles for drawing trees. Specifically, is the problem of computing (not necessarily upward) drawings of trees on a given grid NP-hard? Are there polynomial-time algorithms that can approximate the area within a constant factor?

References

1. Alam, M.J., Samee, M.A.H., Rabbi, M., Rahman, M.S.: Minimum-layer upward drawings of trees. *J. Graph Algorithms Appl.* **14**(2), 245–267 (2010)
2. Bachmaier, C., Matzeder, M.: Drawing unordered trees on k -grids. *J. Graph Algorithms Appl.* **17**(2), 103–128 (2013)
3. Bhatt, S.N., Cosmadakis, S.S.: The complexity of minimizing wire lengths in VLSI layouts. *Inf. Process. Lett.* **25**(4), 263–267 (1987)
4. Biedl, T.: On area-optimal planar graph drawings. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) *ICALP 2014, Part I. LNCS*, vol. 8572, pp. 198–210. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43948-7_17
5. Biedl, T.: Optimum-width upward drawings of trees I: rooted pathwidth. *CoRR* abs/1502.02753 (2015)
6. Biedl, T., Mondal, D.: On upward drawings of trees on a given grid. *CoRR* abs/1708.09515 (2017). <http://arxiv.org/abs/1708.09515>
7. Brunner, W., Matzeder, M.: Drawing ordered $(k - 1)$ -ary trees on k -grids. In: Brandes, U., Cornelsen, S. (eds.) *GD 2010. LNCS*, vol. 6502, pp. 105–116. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-18469-7_10
8. Di Battista, G., Frati, F.: A survey on small-area planar graph drawing (2014), *coRR* report 1410.1006
9. Garey, M., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W H Freeman & Co, New York (1979)
10. Gregori, A.: Unit-length embedding of binary trees on a square grid. *Inf. Process. Lett.* **31**(4), 167–173 (1989)
11. Krug, M., Wagner, D.: Minimizing the area for planar straight-line grid drawings. In: Hong, S.-H., Nishizeki, T., Quan, W. (eds.) *GD 2007. LNCS*, vol. 4875, pp. 207–212. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-77537-9_21
12. Mahaney, S.R.: Sparse complete sets of NP: Solution of a conjecture of Berman and Hartmanis. *J. Comput. Syst. Sci.* **25**(2), 130–143 (1982)
13. Marriott, K., Stuckey, P.J.: NP-completeness of minimal width unordered tree layout. *J. Graph Algorithms Appl.* **8**(2), 295–312 (2004)
14. Mondal, D., Alam, M.J., Rahman, M.S.: Minimum-layer drawings of trees. In: Katoh, N., Kumar, A. (eds.) *WALCOM 2011. LNCS*, vol. 6552, pp. 221–232. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19094-0_23

15. Mondal, D., Nishat, R.I., Rahman, M.S., Alam, M.J.: Minimum-area drawings of plane 3-trees. *J. Graph Algorithms Appl.* **15**(2), 177–204 (2011)
16. Supowit, K.J., Reingold, E.M.: The complexity of drawing trees nicely. *Acta Informatica* **18**, 377–392 (1982)
17. Tamassia, R. (ed.): *Handbook of Graph Drawing and Visualization (Discrete Mathematics and Its Applications)*. Chapman and Hall/CRC, Boca Raton (2014)
18. Trevisan, L.: A note on minimum-area upward drawing of complete and Fibonacci trees. *Inf. Process. Lett.* **57**(5), 231–236 (1996)

Simple Compact Monotone Tree Drawings

Anargyros Oikonomou¹ and Antonios Symvonis²(✉)

¹ School of Electrical and Computer Engineering,
National Technical University of Athens, Athens, Greece

² School of Applied Mathematical and Physical Sciences,
National Technical University of Athens, Athens, Greece
symvonis@math.ntua.gr

Abstract. A monotone drawing of a graph G is a straight-line drawing of G such that every pair of vertices is connected by a path that is monotone with respect to some direction.

Trees, as a special class of graphs, have been the focus of several papers and, recently, He and He [6] showed how to produce a monotone drawing of an arbitrary n -vertex tree that is contained in a $12n \times 12n$ grid.

In this paper, we present a simple algorithm that constructs for each arbitrary tree a monotone drawing on a grid of size at most $n \times n$.

1 Introduction

A *straight-line drawing* Γ of a graph G is a mapping of each vertex to a distinct point on the plane and of each edge to a straight-line segment between the vertices. A path $P = \{p_0, p_1, \dots, p_n\}$ is *monotone* if there exists a line l such that the projections of the vertices of P on l appear on l in the same order as on P . A straight-line drawing Γ of a graph G is *monotone*, if a *monotone* path connects every pair of vertices.

Monotone graph drawing has gained the recent attention of researchers and several interesting results have appeared. Given a planar fixed embedding of a planar graph G , a planar monotone drawing of G can be constructed, but at the cost of some bends on some edges [2]. In the variable embedding setting, we can construct a planar monotone drawing of any planar graph without any bends [8].

One way to find a monotone drawing of a graph is to simply find a monotone drawing of one of its spanning trees. For that reason, the problem of finding monotone drawings of trees has been the subject of several recent papers, starting from the work by Angelini et al. [1] which introduced monotone graph drawings. Angelini et al. [1] provided two algorithms that used ideas from number theory and more specifically Stern-Brocot trees [3, 11], [4, Sect. 4.5]. The first algorithm used a grid of size $O(n^{1.6}) \times O(n^{1.6})$ (BFS-based algorithm) while the second one used a grid of size $O(n) \times O(n^2)$ (DFS-based algorithm). Later, Kindermann

The work of Prof. Symvonis was supported by the iRead H2020 research grant (No. 731724).

et al. [9] provided an algorithm based on Farey sequence (see [4, Sect. 4.5]) that used a grid of size $O(n^{1.5}) \times O(n^{1.5})$. He and He [7] gave an algorithm based on Farey sequence and reduced the required grid size to $O(n^{1.205}) \times O(n^{1.205})$, which was the first result that used less than $O(n^3)$ area. Recently, He and He [5] firstly reduced the grid size for a monotone tree drawing to $O(n \log(n)) \times O(n \log(n))$ and, in a sequel paper, to $O(n) \times O(n)$ [6]. Their monotone tree drawing uses a grid of size at most $12n \times 12n$ which turns out to be asymptotically optimal as there exist trees which require at least $\frac{n}{12} \times \frac{n}{12}$ area [6].

Our Contribution: We provide a simple algorithm that given any n -vertex tree T , outputs a monotone drawing of T on a grid of size $n \times n$. Example drawings of our algorithm appear at Figs. 1, 2, 3, 4 and 5. Our algorithm does not employ number theory techniques but a rather simple weighting method and some simple facts from geometry that can be more analytically expressed. Due to space limitation, some proofs appear in the arXiv version of the paper [10].

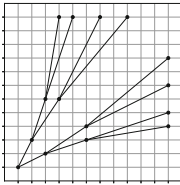


Fig. 1. 3-layer full binary tree (15 nodes).

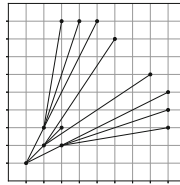


Fig. 2. 2-layer full ternary tree (13 nodes).

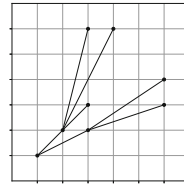


Fig. 3. Tree used in [5,6].

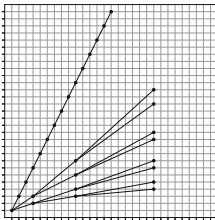


Fig. 4. 3-layer full binary tree plus path (29 nodes).

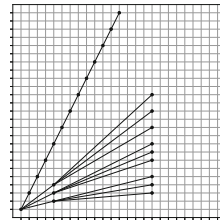


Fig. 5. 2-layer full ternary tree plus path (25 nodes).

2 Definitions and Preliminaries

Let Γ be a drawing of a graph G and (u, v) be an edge from vertex u to vertex v in G . The slope of edge (u, v) , denoted by $slope(u, v)$, is the angle spanned by a counter-clockwise rotation that brings a horizontal half-line starting at u and

directed towards increasing x-coordinates to coincide with the half-line starting at u and passing through v . We consider slopes that are equivalent modulo 2π as the same slope. Observe that $\text{slope}(u, v) = \text{slope}(v, u) - \pi$.

Let T be a tree rooted at a node r . Denote by T_u the subtree of T rooted at a node u . By $|T_u|$ we denote the number of vertices of T_u . In the rest of the paper, we assume that all tree edges are directed away from the root.

In order to simplify the description of our algorithm, we extend the definition of *slope-disjoint* tree drawings given by Angelini et al. [1]. More specifically, a tree drawing Γ of a rooted tree T is called a *non-strictly slope-disjoint* drawing if the following conditions hold:

1. For every node $u \in T$, there exist two angles $a_1(u)$ and $a_2(u)$, with $0 \leq a_1(u) < a_2(u) \leq \pi$ such that for every edge e that is either in T_u or enters u from its parent, it holds that $a_1(u) < \text{slope}(e) < a_2(u)$.
2. For every two nodes $u, v \in T$ such that v is a child of u , it holds that $a_1(u) \leq a_1(v) < a_2(v) \leq a_2(u)$.
3. For every two nodes u_1, u_2 with the same parent, it holds that either $a_1(u_1) < a_2(u_1) \leq a_1(u_2) < a_2(u_2)$ or $a_1(u_2) < a_2(u_2) \leq a_1(u_1) < a_2(u_1)$.

The idea behind the original definition of slope-disjoint tree drawings is that all edges in the subtree T_u as well as the edge entering u from its parent will have slopes that *strictly* fall within the angle range $(a_1(u), a_2(u))$ defined for vertex u . $(a_1(u), a_2(u))$ is called the *angle range of u* with $a_1(u)$ and $a_2(u)$ being its *boundaries*. In our extended definition, we allow for angle ranges of adjacent vertices (parent-child relationship) or sibling vertices (children of the same parent) to share angle range boundaries. Note that replacing the “ \leq ” symbols in our definition by the “ $<$ ” symbol gives us the original definition of Angelini et al. [1] for the slope disjoint tree drawings.

Lemma 1. *Every non-strictly slope-disjoint drawing of a tree T is also a slope-disjoint drawing.*

Theorem 1. [1] *Every slope-disjoint drawing of a tree is monotone.*

Theorem 2. *Every non-strictly slope-disjoint drawing of a tree is monotone.*

Based on geometry, we now prove that it is always possible to identify points on a grid that satisfy several properties with respect to their location.

Lemma 2 [See Fig. 6]. *Consider two angles θ_1, θ_2 with $0 \leq \theta_1 < \theta_2 \leq \frac{\pi}{4}$, and let $d = \lceil \frac{1}{\theta_2 - \theta_1} \rceil$. Then, edge e connecting the origin $(0, 0)$ to point $p = (d, \lfloor \tan(\theta_1)d + 1 \rfloor)$ satisfies $\theta_1 < \text{slope}(e) < \theta_2$.*

Lemma 3 [See Fig. 7]. *Consider angles θ_1, θ_2 with $0 \leq \theta_1 < \theta_2 \leq \frac{\pi}{2}$ and let $d = \lceil \frac{1}{\theta_2 - \theta_1} \rceil$. Then, a grid point p such that the edge e that connects the origin $(0, 0)$ to p satisfies $\theta_1 < \text{slope}(e) < \theta_2$, can be identified as follows:*

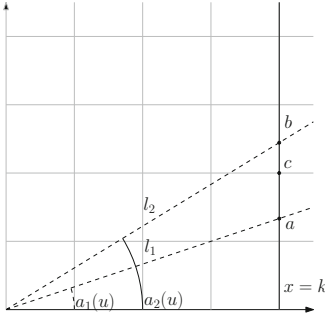


Fig. 6. Geometric representation of Lemma 2.

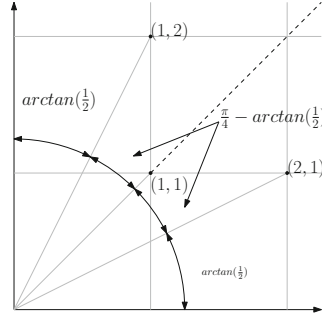


Fig. 7. Point, slopes angular sectors used in Lemma 3.

$$\theta_2 - \theta_1 > \frac{\pi}{4} : \quad p = (1, 1)$$

$$\frac{\pi}{4} \geq \theta_2 - \theta_1 > \arctan(\frac{1}{2}) : \quad \begin{cases} p = (1, 2) & \text{if } \theta_1 \geq \frac{\pi}{4} \\ p = (1, 1) & \text{if } \frac{\pi}{4} > \theta_1 \geq \arctan(\frac{1}{2}) \\ p = (2, 1) & \text{if } \arctan(\frac{1}{2}) > \theta_1 \end{cases}$$

$$\arctan(\frac{1}{2}) \geq \theta_2 - \theta_1 : \quad \begin{cases} p = (d, \lfloor \tan(\theta_1)d + 1 \rfloor) & \text{if } \frac{\pi}{4} \geq \theta_2 > \theta_1 \geq 0 \\ p = (1, 1) & \text{if } \theta_2 > \frac{\pi}{4} > \theta_1 \\ p = (\lfloor \tan(\frac{\pi}{2} - \theta_2)d + 1 \rfloor, d) & \text{if } \theta_2 > \theta_1 \geq \frac{\pi}{4} \end{cases}$$

Moreover, if $p = (x, y)$ is the identified point, it also holds that:

$$\max(x, y) \begin{cases} \leq \frac{\pi}{2} \frac{1}{\theta_2 - \theta_1} & \text{if } \theta_2 - \theta_1 > \arctan(\frac{1}{2}) \\ < \frac{1}{\theta_2 - \theta_1} + 1 & \text{if } \arctan(\frac{1}{2}) \geq \theta_2 - \theta_1 \end{cases}$$

3 Monotone Tree Drawing on an $n \times n$ Grid

Our tree drawing algorithm will produce a non-strictly slope-disjoint tree drawing which, by Theorem 2, is monotone. We make the assumption that the given tree is rooted, otherwise, it can be rooted at any arbitrary node. In order to describe a non-strictly slope-disjoint tree drawing, we need to identify for each vertex u of the tree a grid point to draw u as well as to assign to it two angles $a_1(u), a_2(u)$, with $a_2(u) > a_1(u)$. For every tree vertex, the identified grid point and the two angles should be such that the three properties of the non-strictly slope-disjoint drawing are satisfied.

The basic idea behind our algorithm is to split in a balanced way the angle range $(a_1(u), a_2(u))$ of vertex u to its children based on the size of the subtrees rooted at them. The following lemma formalizes this idea.

Lemma 4. *Let u be a node of the rooted tree T such that we already have assigned values for $a_1(u)$ and $a_2(u)$, with $a_1(u) < a_2(u)$. Let $u_1, u_2, \dots, u_m, m \geq 1$, be the children of u in T . Then, the following assignment of a_1, a_2 for the*

children of u satisfies Property-2 and Property-3 of the non-strictly slope disjoint drawing:

$$a_1(u_i) = \begin{cases} a_1(u) & \text{if } i = 1 \\ a_2(u_{i-1}) & \text{if } 1 < i \leq m \end{cases}$$

$$a_2(u_i) = a_1(u_i) + (a_2(u) - a_1(u)) * \frac{|T_{u_i}|}{|T_u|-1}, \quad 1 \leq i \leq m$$

Observation 1. *If a vertex u has only one child, say u_1 , then the angle assignment strategy of Lemma 4 assigns $a_1(u_1) = a_1(u)$ and $a_2(u_1) = a_2(u)$, which means that the child “inherits” the angle-range of its parent.*

Algorithm 1 describes our monotone tree drawing algorithm. It consists of three steps: Procedure ASSIGNANGLES which assigns angle-ranges to the vertices of the tree according to Lemma 4, Procedure DRAWVERTICES which assigns each tree vertex to a grid point according to Lemma 3 and Procedure BALANCEDTREETEMONOTONEDRAW which assigns the root to point $(0, 0)$ with angle-range $(0, \frac{\pi}{2})$ and initiates the drawing of the tree.

Algorithm 1. Balanced Monotone Tree Drawing algorithm

- 1: **procedure** BALANCEDTREETEMONOTONEDRAW
 - 2: Input: An n -vertex tree T rooted at vertex r .
 - 3: Output: A monotone drawing of T on a grid of size at most $n \times n$.
 - 4: $a_1(r) \leftarrow 0, a_2(r) \leftarrow \frac{\pi}{2}$
 - 5: ASSIGNANGLES($r, a_1(r), a_2(r)$)
 - 6: Draw r at $(0, 0)$
 - 7: DRAWVERTICES(r)
 - 8: **procedure** ASSIGNANGLES(u, a_1, a_2)
 - 9: Input: A vertex u and the boundaries of the angle-range (a_1, a_2) assigned to u .
 - 10: Action: It assigns angle-ranges to the vertices of T_u .
 - 11: **for** each child u_i of u **do**
 - 12: Assign $a_1(u_i), a_2(u_i)$ as described in Lemma 4.
 - 13: ASSIGNANGLES($u_i, a_1(u_i), a_2(u_i)$)
 - 14: **procedure** DRAWVERTICES(u)
 - 15: Input: A vertex u that has already been drawn on the grid.
 - 16: Action: It draws the vertices of T_u .
 - 17: **for** each child u_i of u **do**
 - 18: Find a valid pair (x, y) as in Lemma 3 where $\theta_1 \leftarrow a_1(u), \theta_2 \leftarrow a_2(u)$
 - 19: If u is drawn at (u_x, u_y) , draw u_i at $(u_x + x, u_y + y)$
 - 20: DRAWVERTICES(u_i)
-

Lemma 5. *The drawing produced by Algorithm 1 is monotone.*

Proof. The angle-range assignment satisfies Property-2 and Property-3 of the non-strictly slope disjoint drawing as proved in Lemma 4. In addition, the assignment of the vertices to grid points satisfies Property-1 of the non-strictly slope disjoint drawing as proved in Lemma 3. Thus, the produced drawing by Algorithm 1 is non-strictly slope disjoint and, by Theorem 2, it is monotone. \square

It remains to establish a bound on the grid size required by Algorithm 1. Our proof will use induction on the number of tree vertices having more than one child. The following lemma will be used as the basis of our induction.

Lemma 6. *Let T be an n -vertex rooted tree in which all vertices have at most one child, i.e., T is a path rooted at one of its endpoints. Then, Algorithm 1 draws T in the diagonal of an $n \times n$ grid.*

Lemma 7. *Let T be a rooted tree in which $k > 0$ of its vertices have at least two children. Let u be a vertex with at least two children and, moreover, every other vertex in T_u has at most one child. Let T' be the tree derived by replacing (in T) the subtree T_u by a path of length $|T_u|$. Then, the size of the grid which Algorithm 1 uses in the worst case for the drawing of T is smaller or equal to the size of the grid it uses in the worst case for the drawing of T' .*

Proof. Let u be a vertex as the one stated in the lemma, i.e., in T_u , u is the only vertex having at least two children. Let u_1, u_2, \dots, u_m , $m \geq 2$, be the children of u , and let T_{u_i} be the subtree rooted at u_i . Note that each T_{u_i} is a path. From Observation 1, we recall that for each node in T_{u_i} , the assigned values for a_1 and a_2 by Algorithm 1 will be the same as $a_1(u_i)$ and $a_2(u_i)$. Let $\phi(u) = a_2(u) - a_1(u)$. We consider two subcases based on whether $\arctan(\frac{1}{2}) \geq \phi(u)$ or not.

Case-1: $\arctan(\frac{1}{2}) \geq \phi(u)$. Since Algorithm 1 performs its angle-range assignment based on Lemma 4, for each child of u we have that $\phi(u_i) = a_2(u_i) - a_1(u_i) = \frac{|T_{u_i}|}{|T_u|-1} \phi(u)$. Observe that it also holds that $\phi(u_i) \leq \arctan(\frac{1}{2})$.

A node in T_{u_i} is drawn, based on Lemma 3 where $\theta_1 \leftarrow a_1(u_i)$ and $\theta_2 \leftarrow a_2(u_i)$, in a grid of length at most $\frac{1}{\theta_2(u_i) - \theta_1(u_i)} + 1$ if its parent is considered to be drawn at the origin. So, the length of the total grid that is used for the drawing of path T_{u_i} is at most: $|T_{u_i}|(\frac{1}{\theta_2(u_i) - \theta_1(u_i)} + 1) = |T_{u_i}| \frac{1}{\frac{|T_{u_i}|}{|T_u|-1} \phi(u)} + |T_{u_i}| = \frac{|T_u|-1}{\phi(u)} + |T_{u_i}| \leq \frac{|T_u|-1}{\phi(u)} + |T_u| - 1 = (|T_u| - 1)(\frac{1}{\phi(u)} + 1)$.

The last term is the maximum grid length dictated by Lemma 3 for the drawing of a path of size $|T_u|$ with $\theta_1 \leftarrow a_1(u)$ and $\theta_2 \leftarrow a_2(u)$. Note also that in Algorithm 1 the largest grid devoted to any of T_{u_i} , $1 \leq i \leq m$, determines the grid size of the drawing of T_u since the subtrees rooted at children of u are drawn completely inside non-overlapping (but possibly touching) angular sectors. The above statement holds because all the grids that will be used for the subtrees have the same origin (u) and all angular sectors lies in the first quadrant since Algorithm 1 assigns root with angle-range $(0, \frac{\pi}{2})$. So, the grid size that is used in the worst case for the drawing of T_u by Algorithm 1 is smaller or equal to that used by it in the worst case for the drawing of a path of length $|T_u|$. Thus, the size of the grid which Algorithm 1 uses in the worst case for the drawing of T is smaller or equal to the size of the grid it uses in the worst case for the drawing of T' .

Case-2: $\phi(u) > \arctan(\frac{1}{2})$. Let $\phi(u_i) = a_2(u_i) - a_1(u_i)$, $1 \leq i \leq m$. We consider two subcases based on whether $\arctan(\frac{1}{2}) \geq \phi(u_i)$ or not.

Case-2a: $\arctan(\frac{1}{2}) \geq \phi(u_i)$. A node in T_{u_i} is drawn, based on Lemma 3 where $\theta_1(u_i) \leftarrow a_1(u_i)$ and $\theta_2(u_i) \leftarrow a_2(u_i)$, in a grid of length at most $\frac{1}{\phi(u_i)} + 1 < \frac{\pi}{2} \frac{1}{\phi(u_i)}$, assuming that its parent is drawn at the origin. The last inequality holds for $\phi(u_i) \leq \frac{\pi-2}{2}$, and so it also holds for $\phi(u_i) \leq \arctan(\frac{1}{2}) < \frac{\pi-2}{2}$. So, the maximum length of the total grid that is used for the drawing of path T_{u_i} is at most: $|T_{u_i}| \frac{\pi}{2} \frac{1}{\phi(u_i)} = |T_{u_i}| \frac{\pi}{2} \frac{1}{\frac{|T_{u_i}|}{|T_u|-1} \phi(u)} = \frac{\pi}{2} \frac{|T_u|-1}{\phi(u)}$.

Case-2b: $\phi(u_i) > \arctan(\frac{1}{2})$. A node in T_{u_i} is drawn, based on Lemma 3 where $\theta_1(u_i) \leftarrow a_1(u_i)$ and $\theta_2(u_i) \leftarrow a_2(u_i)$, in a grid of length at most $\frac{\pi}{2} * \frac{1}{\phi(u_i)}$, assuming that its parent is drawn at the origin. So, the maximum length of the total grid that is used for the drawing of path T_{u_i} is: $|T_{u_i}| \frac{\pi}{2} \frac{1}{\phi(u_i)} = |T_{u_i}| \frac{\pi}{2} \frac{1}{\frac{|T_{u_i}|}{|T_u|-1} \phi(u)} = \frac{\pi}{2} \frac{|T_u|-1}{\phi(u)}$.

The last term in both subcases is the maximum grid length dictated by Lemma 3 for the drawing of a path of size $|T_u|$ with $\theta_1 \leftarrow a_1(u)$ and $\theta_2 \leftarrow a_2(u)$ where $\phi(u) > \arctan(\frac{1}{2})$. So, the drawing of T_u uses in the worst case a grid length that is smaller or equal to that used in the worst case for the drawing of a path of length $|T_u|$, when both drawings are done by Algorithm 1. Thus, Algorithm 1 uses in the worst case for the drawing of T a grid of size smaller or equal to the one used in the worst case for the drawing of T' . \square

Theorem 3. *Given a rooted Tree T , Algorithm 1 produces a monotone grid drawing using a grid of size at most $n \times n$.*

Proof. The monotonicity of the drawing follows directly from Lemma 5. By repeatedly applying Lemma 7, we get that in the worst case the drawing of T uses a grid length that is smaller or equal to the one used in the worst case for the drawing of a path of length $|T|$, when both drawings are done by Algorithm 1. By Lemma 6, we get that the used grid is of size at most $n \times n$. \square

References

1. Angelini, P., Colasante, E., Di Battista, G., Frati, F., Patrignani, M.: Monotone drawings of graphs. *J. Graph Algorithms Appl.* **16**(1), 5–35 (2012)
2. Angelini, P., Didimo, W., Kobourov, S., Mchedlidze, T., Roselli, V., Symvonis, A., Wismath, S.: Monotone drawings of graphs with fixed embedding. *Algorithmica* **71**(2), 233–257 (2015)
3. Brocot, A.: Calcul des rouages par approximation, nouvelle methode. *Revue Chronometrique* **6**, 186–194 (1860)
4. Graham, R.L., Knuth, D.E., Patashnik, O.: *Concrete Mathematics: A Foundation for Computer Science*, 2nd edn. Addison-Wesley Longman Publishing Co. Inc., Boston (1994)
5. He, D., He, X.: Nearly optimal monotone drawing of trees. *Theor. Comput. Sci.* **654**, 26–32 (2016)
6. He, D., He, X.: Optimal monotone drawings of trees. *CoRR* abs/1604.03921 (2016)

7. He, X., He, D.: Compact monotone drawing of trees. In: Xu, D., Du, D., Du, D. (eds.) COCOON 2015. LNCS, vol. 9198, pp. 457–468. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21398-9_36
8. Hossain, M.I., Rahman, M.S.: Good spanning trees in graph drawing. *Theor. Comput. Sci.* **607**, 149–165 (2015)
9. Kindermann, P., Schulz, A., Spoerhase, J., Wolff, A.: On monotone drawings of trees. In: Duncan, C., Symvonis, A. (eds.) GD 2014. LNCS, vol. 8871, pp. 488–500. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45803-7_41
10. Oikonomou, A., Symvonis, A.: Simple compact monotone tree drawings. CoRR abs/1708.09653v2 (2017). <http://arxiv.org/abs/1708.09653v2>
11. Stern, M.: Ueber eine zahlentheoretische funktion. *Journal fur die reine und angewandte Mathematik* **55**, 193–220 (1858)

Visualizing Co-phylogenetic Reconciliations

Tiziana Calamoneri¹, Valentino Di Donato², Diego Mariottini²,
and Maurizio Patrignani²(✉)

¹ Computer Science Department, University of Rome “Sapienza”, Rome, Italy

² Engineering Department, Roma Tre University, Rome, Italy

patrigna@dia.uniroma3.it

Abstract. We introduce a hybrid metaphor for the visualization of the reconciliations of co-phylogenetic trees, that are mappings among the nodes of two trees. The typical application is the visualization of the co-evolution of hosts and parasites in biology. Our strategy combines a space-filling and a node-link approach. Differently from traditional methods, it guarantees an unambiguous and ‘downward’ representation whenever the reconciliation is time-consistent (i.e., meaningful). We address the problem of the minimization of the number of crossings in the representation, by giving a characterization of planar instances and by establishing the complexity of the problem. Finally, we propose heuristics for computing representations with few crossings.

1 Introduction

Producing readable and compact representations of trees has a long tradition in the graph drawing research field. In addition to the standard node-link diagrams, which include layered trees, radial trees, hv-drawings, etc., trees can be visualized via the so-called space-filling metaphors, which include circular and rectangular treemaps, sunbursts, icicles, sunrays, icerays, etc. [14, 15].

Unambiguous and effective representation of co-phylogenetic trees, that are pairs of phylogenetic trees with a mapping among their nodes, is needed in biological research. A *phylogenetic tree* is a full rooted binary tree (each node has zero or two children) representing the evolutionary relationships among related organisms. Biologists who study the co-evolution of species, such as hosts and parasites, start with a host phylogenetic tree H , a parasite tree P , and a mapping function φ (not necessarily injective nor surjective) from the leaves of P to the leaves of H . The triple $\langle H, P, \varphi \rangle$, called *co-phylogenetic tree*, is traditionally represented with a tanglegram drawing, that consists of a pair of plane trees whose leaves are connected by straight-line edges [2–4, 9, 10, 12, 16]. However, a tanglegram only represents the input of a more complex process that aims at computing a mapping γ , called *reconciliation*, that extends φ and maps all the parasite nodes onto the host nodes.

This research was partially supported by MIUR project “MODE – MORphing graph Drawings Efficiently”, prot. 20157EFM5C.001 and by *Sapienza* University of Rome project “Combinatorial structures and algorithms for problems in co-phylogeny”.

Given H , P , and φ , a great number of different reconciliations are possible. Some of them can be discarded, since they are not consistent with time (i.e. they induce contradictory constraints on the periods of existence of the species associated to internal nodes). The remaining reconciliations are generally ranked based on some quality measure and only the optimal ones are considered. Even so, optimal reconciliations are so many that biologists have to perform a painstaking manual inspection to select those that are more compatible with their understanding of the evolutionary phenomena.

In this paper we propose a new and unambiguous metaphor to represent reconciliations of co-phylogenetic trees (Sect. 3). The main idea is that of representing H in a suitable space-filling style and of using a traditional node-link style to represent P . This is the first representation guaranteeing the downwardness of P when time-consistent (i.e., meaningful) reconciliations are considered. In order to pursue readability, we study the number of crossings that are introduced in the drawing of tree P (tree H is always planar): on the one hand, in Sect. 4 we characterize planar reconciliations, on the other hand, we show in Sect. 5 that reducing the number of crossings in the representation of the reconciliations is NP-complete. Finally, we propose heuristics to produce drawings with few crossings (Sect. 6) and experimentally show their effectiveness and efficiency (Sect. 7). Details and full proofs can be found in [5].

2 Background

In this paper, whenever we mention a tree T , we implicitly assume that it is a *full rooted binary tree* with node set $\mathcal{V}(T)$ and arc set $\mathcal{A}(T)$, and that arcs are oriented away from the root $r(T)$ down to the set of leaves $\mathcal{V}_L(T) \subset \mathcal{V}(T)$ (see [5] for formal definitions). The *lowest common ancestor* of two nodes $u, v \in \mathcal{V}(T)$, denoted $lca(u, v)$, is the last common node of the two directed paths leading from $r(T)$ to u and v . Two nodes u and v are *comparable* if $lca(u, v) \in \{u, v\}$, otherwise they are *incomparable*.

A *tanglegram* $\langle T_1, T_2, G \rangle$ generalizes a co-phylogenetic tree and consists of two generic rooted trees T_1 and T_2 and a bipartite graph $G = (\mathcal{V}_L(T_1), \mathcal{V}_L(T_2), E)$ among their leaves. In a *tanglegram drawing* of $\langle T_1, T_2, G \rangle$: (i) tree T_1 is planarly drawn above a horizontal line l_1 with its arcs pointing downward and its leaves on l_1 , (ii) tree T_2 is planarly drawn below a horizontal line l_2 , parallel to l_1 , with its arcs pointing upward and its leaves on l_2 , and (iii) edges of G , called *tangles*, are straight-line segments drawn in the horizontal stripe bounded by l_1 and l_2 . A decade-old literature is devoted to tanglegram drawings (see e.g. [2–4, 9, 10, 12, 16]). Finding a tanglegram drawing that minimizes the number of crossings among the edges in E is known to be NP-complete, even if the trees are binary trees or if the graph G is a matching [10].

A *reconciliation* of the co-phylogenetic tree $\langle H, P, \varphi \rangle$ is a mapping $\gamma : \mathcal{V}(P) \rightarrow \mathcal{V}(H)$ that satisfies the following properties: (i) for any $p \in \mathcal{V}_L(P)$, $\gamma(p) = \varphi(p)$, that is, γ extends φ , (ii) for any arc $(p_i, p_j) \in \mathcal{A}(P)$, $lca(\gamma(p_i), \gamma(p_j)) \neq \gamma(p_j)$, that is, a child p_j of p_i cannot be mapped to an ancestor of $\gamma(p_i)$ and (iii)

for any $p \in V \setminus \mathcal{V}_L(P)$ with children p_1 and p_2 , $lca(\gamma(p), \gamma(p_1)) = \gamma(p)$ or $lca(\gamma(p), \gamma(p_2)) = \gamma(p)$, that is, at least one of the two children is mapped in the subtree rooted at $\gamma(p)$.

The set of all reconciliations of $\langle H, P, \varphi \rangle$ is denoted $\mathcal{R}(H, P, \varphi)$.

Four types of events may take place in a reconciliation (see formal definitions in [5]): *co-speciation*, when both the host and the parasite speciate; *duplication*, when the parasite speciates (but not the host) and both parasite children remain associated with the host; *loss*, when the host speciates but not the parasite, leading to the loss of the parasite in one of the two host children; and *host-switch*, when the parasite speciates and one child remains with the current host while the other child jumps to an incomparable host.

Each of the above events is usually associated with a penalty and the minimum cost reconciliations are searched (they can be computed with polynomial delay [8, 21]). However, only reconciliations not violating obvious temporal constraints are of interest. A reconciliation γ is *time-consistent* if there exists a linear ordering π of the parasites $\mathcal{V}(P)$ such that: (i) for each arc $(p_1, p_2) \in \mathcal{A}(P)$, $\pi(p_1) < \pi(p_2)$; (ii) for each pair $p_1, p_2 \in \mathcal{V}(P)$ such that $\pi(p_1) < \pi(p_2)$, $\gamma(p_2)$ is not a proper ancestor of $\gamma(p_1)$. Recognizing time-consistent reconciliations is a polynomial task [8, 18, 21], while producing exclusively time-consistent reconciliations is NP-complete [13, 19]. This is why usually time-inconsistent reconciliations are filtered out in a post-processing step [1].

The available tools to compute reconciliations adopt three main conventions to represent them. The simplest strategy, schematically represented in Fig. 1(a), represents the two trees by adopting the traditional node-link metaphor, where the nodes of P are drawn close to the nodes of H they are associated to. Unfortunately, when several parasite nodes are associated to the same host node, the drawing becomes cluttered and the attribution of parasite nodes to host nodes becomes unclear. Further, even if P was drawn without crossings (tree H always is), the overlapping of the two trees produces a high number of crossings (see [5]).

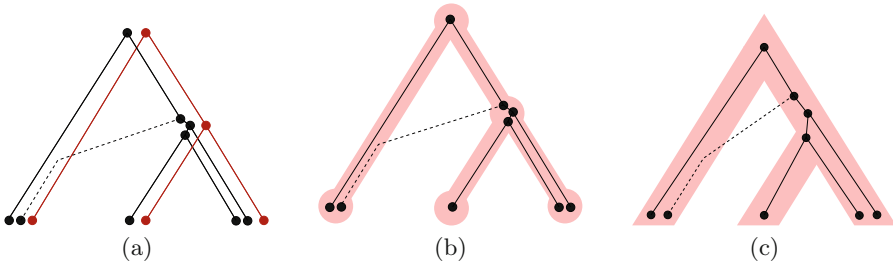


Fig. 1. Three visualization strategies for representing co-phylogenetic trees.

An alternative strategy (Fig. 1(b)) consists in representing H as a background shape, such that its nodes are shaded disks and its arcs are thick pipes, while P is contained in H and drawn in the traditional node-link style. This strategy is

used, for example, by *CophyTrees* [1], the viewer associated with the Eucalypt tool [8]. The representation is particularly effective, as it is unambiguous and crossings between the two trees are strongly reduced, but it is still cluttered when a parasite subtree has to be squeezed inside the reduced area of a host node (see [5]).

Finally, some visualization tools adopt the strategy of keeping the containment metaphor while only drawing thick arcs of H and omitting host nodes (Fig. 1(c)). This produces a node-link drawing of the parasite tree drawn inside the pipes representing the host tree. Examples include *Primetv* [17] and *SylvX* [6]—see [5]. Also this strategy is sometimes ambiguous, since it is unclear how to attribute parasites to hosts.

3 A New Model for the Visualization of Reconciliations

Inspired by recent proposals of adopting space-filling techniques to represent biological networks [20], and with the aim of overcoming the limitations of existing visualization strategies, we introduce a new hybrid metaphor for the representation of reconciliations. A space-filling approach is used to represent H , while tree P maintains the traditional node-link representation. The reconciliation is unambiguously conveyed by placing parasite nodes inside the regions associated with the hosts they are mapped to.

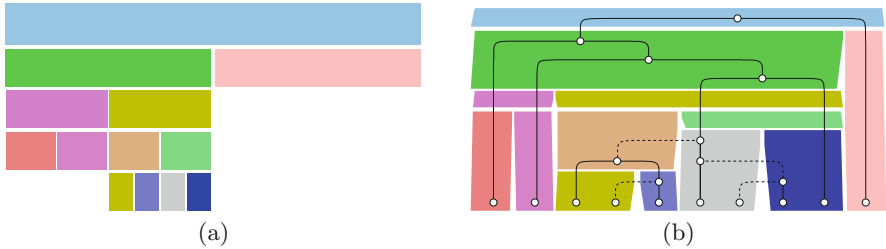


Fig. 2. (a) An icicle. (b) The representation adopted for trees H and P .

More specifically, the representation of tree H is a variant of a representation known in the literature with the name of *icicle* [11]. An *icicle* is a space-filling representation of hierarchical information in which nodes are represented by rectangles and arcs are represented by the contact of rectangles, such that the bottom side of the rectangle representing a node touches the top sides of the rectangles representing its children (see Fig. 2(a)). In our model, in order to contain parasite subtrees of different depths, we allow rectangles of different height. Also we force all leaves of H (i.e. present-day hosts) to share the same bottom line that intuitively represents current time.

Formally, an *HP-drawing* $\Gamma(\gamma)$ of $\gamma \in \mathcal{R}(H, P, \varphi)$ is the simultaneous representation of H and P as follows. Tree H is represented in a space-filling fashion

such that: (1) nodes of H are represented by internally disjoint rectangles that cover the drawing area; the rectangle corresponding to the root of H covers the top border of the drawing area while the rectangles corresponding to the leaves of H touch the bottom border of the drawing area with their bottom sides; and (2) arcs of H are represented by the vertical contact of rectangles, the upper rectangle being the parent and the lower rectangle being the child. Conversely, tree P is represented in a node-link style such that: (1) each node $p \in \mathcal{V}(P)$ is drawn as a point in the plane inside the representation of the rectangle corresponding to node $\gamma(p)$; and (2) each arc $(p_1, p_2) \in \mathcal{A}(P)$ is drawn as a vertical segment if p_1 and p_2 have the same x -coordinate; otherwise, it is drawn as a horizontal segment followed by a vertical segment.

It can be assumed that an HP-drawing only uses integer coordinates. In particular the corners of the rectangles representing the nodes of H could exclusively use even coordinates and the nodes of P could exclusively use odd coordinates.

Graphically, since the icicle represents a binary tree, we give the rectangles a slanted shape in order to ease the visual recognition of the two children of each node (see Fig. 2(b)). Also, the bend of an arc of P is a small circular arc.

We say that HP-drawing $\Gamma(\gamma)$ is *planar* if no pair of arcs of P intersect except, possibly, at a common endpoint, and that it is *downward* if, for each arc $(p_1, p_2) \in \mathcal{A}(P)$, parasite p_1 has a y -coordinate greater than that of parasite p_2 .

4 Planar Instances and Reconciliations

In this section we characterize the reconciliations that can be planarly drawn, showing that a time-consistent reconciliation is planar if and only if the corresponding co-phylogenetic tree admits a planar tanglegram drawing.

Theorem 1. *Given a co-phylogenetic tree $\langle H, P, \varphi \rangle$, the following statements are equivalent: (1) $\langle H, P, \varphi \rangle$ admits a planar tanglegram drawing Δ . (2) Every time-consistent reconciliation $\gamma \in \mathcal{R}(H, P, \varphi)$ admits a planar downward HP-drawing $\Gamma(\gamma)$.*

Sketch of proof. First, we prove that (2) implies (1). Consider a planar drawing $\Gamma(\gamma)$ of $\gamma \in \mathcal{R}(H, P, \varphi)$ and let l be the horizontal line passing through the bottom border of $\Gamma(\gamma)$. Observe that the leaves of P lie above l . Construct a tanglegram drawing Δ of $\langle H, P, \varphi \rangle$ as follows: (a) Draw H by placing each node $h \in \mathcal{V}(H)$ in the center of the rectangle representing h in $\Gamma(\gamma)$ and by representing each arc $a \in \mathcal{A}(H)$ as a suitable curve between its incident nodes; (b) draw P in Δ as a mirrored drawing with respect to l of the drawing of P in $\Gamma(\gamma)$; (c) connect each leaf $p \in L(P)$ to the host $\gamma(p)$ with a straight-line segment. It is immediate that Δ is a tanglegram drawing of $\langle H, P, \varphi \rangle$ and that it is planar whenever $\Gamma(\gamma)$ is.

Proving that (1) implies (2) is more laborious. Let Δ be a planar tanglegram drawing of $\langle H, P, \varphi \rangle$. We construct a drawing $\Gamma(\gamma)$ of the given time-consistent reconciliation $\gamma \in \mathcal{R}(H, P, \varphi)$ as follows. First, insert into the arcs of P dummy

nodes of degree two to represent losses, obtaining a new tree P' . Since γ is time-consistent, consider any ordering π' of $\mathcal{V}(P')$ consistent with H . Remove from π' the leaves of P and renumber the remaining nodes obtaining a new ordering π from 1 to $|\mathcal{V}(P') - \mathcal{V}_L(P')|$. Regarding y -coordinates: all the leaves of P' have y -coordinate 1, that is, they are placed at the bottom of the drawing, while each internal node $p \in \mathcal{V}(P') \setminus \mathcal{V}_L(P')$ has y -coordinate $2\pi(p) + 1$. Regarding x -coordinates: each leaf $p \in \mathcal{V}_L(P)$ has x -coordinate $2\sigma(p) + 1$, where $\sigma(p)$ is the left-to-right order of the leaves of T_2 in Δ . The x -coordinate of an internal node p of P is copied from one of its children p_1 or p_2 , arbitrarily chosen if none of them is connected by a host-switch, the one (always present) that is not connected by a host-switch otherwise.

Let h be a node of $\mathcal{V}(H)$; rectangle R_h , representing h in Γ , has the minimum width that is sufficient to span all the parasites contained in the subtree $T_h(H)$ of H rooted at h (hence, it spans the interval $[x_{min} - 1, x_{max} + 1]$, where x_{min} and x_{max} are the minimum and maximum x -coordinates of a parasite contained in $T_h(H)$, respectively). The top border of R_h has y -coordinate $y_{MIN} - 1$, where y_{MIN} is the minimum y -coordinate of a parasite node contained in the parent of h . The bottom border of R_h is $y_{min} - 1$, where y_{min} is the minimum y -coordinate of a parasite node contained in h .

The proof concludes by showing that the obtained representation $\Gamma(\gamma)$ is planar and downward [5]. □

We remark that a statement analogous to the one of Theorem 1 can be proved also for the visualization strategy schematically represented in Fig. 1(b) and adopted, for example, by *CophyTrees* [1].

The algorithm we actually implemented, called **PlanarDraw**, is a refinement of the one described in the proof of Theorem 1. It assigns to the parent parasite an x -coordinate that is intermediate between those of the children whenever both

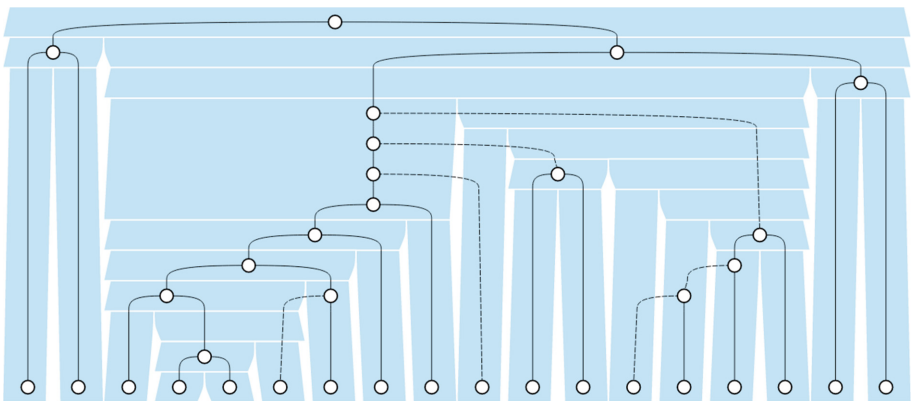


Fig. 3. A planar HP-drawing of a reconciliation of the co-phylogenetic tree of Pelican & Lice (MP) computed by Algorithm **PlanarDraw**.

children are not host-switches and it produces a more compact representation with respect to the y -axis (see Figs. 2(b) and 3).

5 Minimizing the Number of Crossings

In this section we focus on non-planar instances and prove that computing an HP-drawing of a reconciliation with the minimum number of crossings is NP-complete. Given a reconciliation $\gamma \in \mathcal{R}(H, P, \varphi)$ and a constant k , we consider the decision problem RECONCILIATION LAYOUT (RL) that asks whether there exists an HP-drawing of γ that has at most k crossings. We prove that RL is NP-hard by reducing to it the NP-complete problem TWO-TREES CROSSING MINIMIZATION (TTCM) [10]. The input of TTCM consists of two binary trees T_1 and T_2 , whose leaf sets are in one-to-one correspondence, and a constant k . The question is whether T_1 and T_2 admit a tanglegram drawing with at most k crossings among the tangles. In [4] it is shown that TTCM remains NP-complete even if the input trees are two complete binary trees of height h (hence, with 2^h leaves). We reduce this latter variant to RL.

Theorem 2. *Problem RL is NP-complete.*

Sketch of proof. Problem RL is in NP by exploring all possible HP-drawings of γ . Let $I_{\text{TTCM}} = \langle T_1, T_2, \psi, k \rangle$ be an instance of TTCM, where T_1 and T_2 are complete binary trees of height h , ψ is a one-to-one mapping between $\mathcal{V}_L(T_1)$ and $\mathcal{V}_L(T_2)$, and k is a constant. We show how to build an equivalent instance $I_{\text{RL}} = \langle \gamma \in \mathcal{R}(H, P, \varphi), k' \rangle$ of RL.

First we introduce a gadget, called ‘sewing tree’, that will help in the definition of our instance. A *sewing tree* is a subtree of the parasite tree whose nodes are alternatively assigned to two host leaves h_1 and h_2 as follows. A single node p_0 with $\gamma(p_0) = h_2$ is a sewing tree S_0 of size 0 and root p_0 . Let S_m be a sewing tree of size m and root p_m such that $\gamma(p_m) = h_2$ ($\gamma(p_m) = h_1$, respectively). In order to obtain S_{m+1} we add a node p_{m+1} with $\gamma(p_{m+1}) = h_1$ ($\gamma(p_{m+1}) = h_2$, respectively) and two children, p_m and p'_m , with $\gamma(p'_m) = h_1$ ($\gamma(p'_m) = h_2$, respectively). See Fig. 4 for examples of sewing trees. Intuitively, a sewing tree has the purpose of making costly from the point of view of the

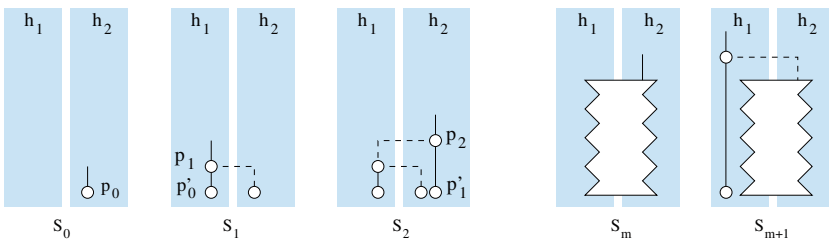


Fig. 4. Sewing trees S_0, S_1, S_2 , and S_{m+1} obtained from S_m .

number of crossings the insertion of a host node h_3 between hosts h_1 and h_2 , whenever h_3 contains several vertical arcs of P towards leaves of the subtree rooted at h_3 .

Nodes $r(H), h_1, h_2, \dots, h_8$ of the host tree H and their relationships are depicted in Fig. 5. Rooted at h_5 and h_8 we have two complete binary trees of height h . Intuitively, these two subtrees of H correspond to T_1 and T_2 , respectively (they are drawn filled green and filled pink in Fig. 5). Hence, the leaves $l_{1,1}, l_{1,2}, \dots, l_{1,2^h}$ of T_1 are associated to the leaves $h_{1,1}, h_{1,2}, \dots, h_{1,2^h}$ of the subtree rooted at h_5 , and, similarly, the leaves $l_{2,1}, l_{2,2}, \dots, l_{2,2^h}$ of T_2 are associated to the leaves $h_{2,1}, h_{2,2}, \dots, h_{2,2^h}$ of the subtree rooted at h_8 .

The root $r(P)$ of P has $\gamma(r(P)) = r(H)$. One child of $r(P)$ is the root of a sewing tree between h_3 and h_6 . The other child p_1 , with $\gamma(p_1) = r(H)$, has one child that is the root of a sewing tree between h_3 and h_7 , and one child p_2 , with $\gamma(p_2) = h_2$. Parasite p_2 is the root of a complete binary tree T_h of height h , whose internal nodes are assigned to h_2 , while the leaves are assigned to h_3 . Each one of the 2^h leaves of T_h is associated with a tangle of the instance I_{TTCM} . Namely, suppose $e = (l_{1,i}, l_{2,j})$ is a tangle edge in the instance I_{TTCM} . Then, an arbitrary leaf p_e of T_h is associated with e . Node p_e has children $p_{1,i}$, with $\gamma(p_{1,i}) = h_{1,i}$, and p'_e , with $\gamma(p'_e) = h_3$. Node p'_e , in turn, has children $p_{2,j}$, with $\gamma(p_{2,j}) = h_{2,j}$, and p''_e , with $\gamma(p''_e) = h_3$. Finally, we pose $k' = k + 2^h \cdot (2^h - 1)$.

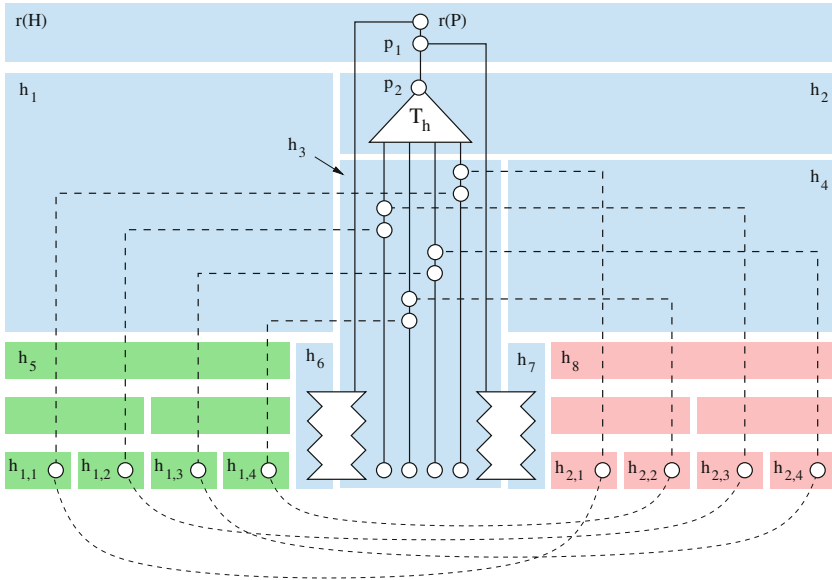


Fig. 5. The construction of the instance of RL starting from an instance of TTCM with $h = 2$. Filled green and pink are the subtrees of H whose embeddings correspond to the embeddings of T_1 and T_2 , respectively. (Color figure online)

The proof concludes by showing that instance I_{TTCM} is a yes instance of TTCM if and only if instance I_{RL} is a yes instance of RL [5]. \square

Since in the proof of Theorem 2 a key role is played by host-switch arcs, one could wonder whether an instance without host-switches is always planar. This is not the case: for any non-planar time-consistent reconciliation $\gamma \in \mathcal{R}(H, P, \varphi)$, there exists a time-consistent reconciliation $\gamma_r \in \mathcal{R}(H, P, \varphi)$ that maps all internal nodes of P to $r(H)$ and that has no host-switch. If the absence of host-switches could guarantee planarity, γ_r would be planar and, by Theorem 1, also γ would be planar, leading to a contradiction. Indeed, it is not difficult to construct reconciliations without host-switches and not planar [5].

6 Heuristics for Drawing Reconciliations with Few Crossings

Theorem 2 shows that a drawing of a reconciliation with the minimum number of crossings cannot be efficiently found. For this reason, we propose two heuristics aiming at producing HP-drawings with few crossings (Fig. 6 shows two examples of non-planar HP-drawings produced by the heuristics). In the following we will briefly describe them.

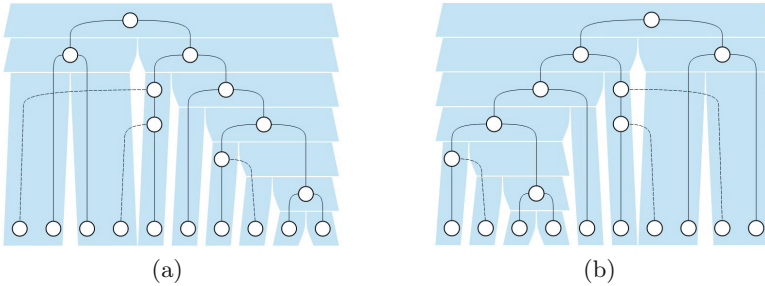


Fig. 6. (1) An HP-drawing of a reconciliation of Gopher & Lice drawn by `SearchMaximalPlanar`. (2) The same instance drawn by `ShortenHostSwitch`.

6.1 Heuristic `SearchMaximalPlanar`

This heuristic is based on the strategy of first drawing a large planar sub-instance and then adding non-planar arcs. We hence construct a maximal planar subgraph G_{pl} of tanglegram $\langle H, P, \varphi \rangle$ by adding to it one by one the following objects: (i) all nodes of H and of P ; (ii) all arcs of H ; (iii) edge $(r(H), r(P))$; (iv) for each $l_p \in \mathcal{V}_L(P)$, edge $(l_p, \varphi(l_p))$; (v) for each $p \in \mathcal{V}(P) \setminus \mathcal{V}_L(P)$, edge (p, p') , where p' is any child of p that is not a host-switch, while the arc from p to the sibling of p' is added to a set `missingArcs`; (vi) all arcs from `missingArcs` that is possible to add without introducing crossings (all arcs that have not been inserted in G_{pl}

are stored in a set of `non-planarArcs`). A planar embedding of the graph G_{pl} is used as input for Algorithm `PlanarDraw` so obtaining a planar drawing of part of reconciliation γ ; arcs in `non-planarArcs` are added in a post-processing step.

6.2 Heuristic ShortenHostSwitch

This heuristic is based on the observation that ‘long’ host-switch arcs are more likely to cause crossings than ‘short’ ones. Hence, this heuristic searches for an embedding of H that reduces the distance between the end-nodes of host-switch arcs of P . To do this, as a preliminary step, `ShortenHostSwitch` chooses the embedding of H with a preorder traversal as follows. Let $v \in \mathcal{V}(H)$ be the current node of the traversal. Consider the set of nodes of H that are ancestors or descendants of v . The removal of this set would leave two connected components, one on the left, denoted $V_{v,left}(H) \subseteq \mathcal{V}(H)$ and one on the right, denoted $V_{v,right}(H) \subseteq \mathcal{V}(H)$. Denote by $V_{v,left}(P)$ ($V_{v,right}(P)$, respectively) the set of parasite nodes mapped to some node in $V_{v,left}(H)$ ($V_{v,right}(H)$, respectively). Moreover, denote by $V_v(P) \subseteq \mathcal{V}(P)$ the set of the parasite nodes mapped to the subtree of H rooted at v .

If $v \in \mathcal{V}_L(H)$ no embedding choice has to be taken for v . Otherwise let v_1 and v_2 be its children. For $i \in \{1, 2\}$ and $X \in \{left, right\}$ compute the number $h_{v_i, X}$ of the host-switch arcs from $V_{v_i}(P)$ to $V_{v, X}(P)$ or vice versa. If $h_{1, right} + h_{2, left} > h_{2, right} + h_{1, left}$ then v_1 is embedded as the right child and v_2 as the left child of v , otherwise v_2 will be the right child and v_1 the left child.

Observe that the sets $V_{v, left}(P)$ and $V_{v, right}(P)$ can be efficiently computed while descending H . Namely, we start with $V_{r(H), left}(P) = V_{r(H), right}(P) = \emptyset$ and, supposing v_l and v_r are chosen to be the left and right children of v , respectively, we set $V_{v_l, left}(P) = V_{v, left}(P)$, $V_{v_l, right}(P) = V_{v, r}(P) \cup V_{v_r, left}(P)$, $V_{v_r, left}(P) = V_{v, left}(P) \cup V_{v_l, right}(P)$, and $V_{v_r, right}(P) = V_{v, right}(P)$.

It remains to describe how `ShortenHostSwitch` places parasite nodes inside the representation of host nodes. First, we temporarily assign to each node $p \in \mathcal{V}(P)$ the lower x - and y -coordinates inside $\gamma(p)$ (observe that all nodes mapped to the same host are overlapped). For the leaves $\mathcal{V}(P)$ the temporary y -coordinate is definitive and only the x -coordinate has to be decided. We order the parasite leaves p_1, p_2, \dots, p_k inside each host leaf v_k as follows. We divide the leaves into two sets $L_{v, left}(P)$ and $L_{v, right}(P)$, where $L_{v, left}(P)$ contains the leaves associated with v that have a parent with lower x -coordinates and $L_{v, right}(P)$ contains the remaining leaves associated with v . We order the set $L_{v, left}(P)$ ($L_{v, right}(P)$, respectively) ascending (descending, respectively) based on the y -coordinates of their parents. We place the set $L_{v, left}(P)$ and then the $L_{v, right}(P)$ inside v according to their orderings. Once the leaves of P have been placed, the remaining internal nodes of P are placed according to the same algorithm used for planar instances by `PlanarDraw`.

7 Experimental Evaluation

We collected standard co-phylogenetic tree instances from the domain literature. Table 1 shows their properties.

Table 1. The co-phylogenetic trees used to generate the datasuite.

Instance	Acronym	# hosts	# par.	Planar
Caryophyllaceae & Microbotryum [21]	CM	35	39	No
Stinkbugs & Bacteria [21]	SB	27	23	Yes
Encyrtidae & Coccidae [8]	EC	13	19	Yes
Fishes & Dactylogyrus [8]	FD	39	101	No
Gopher & Lice [8]	GL	15	19	No
Seabirds & Chewing Lice [8]	SC	21	27	No
Rodents & Hantaviruses [8]	RH	67	83	No
Smut Fungi & Caryophyll. plants [8]	SFC	29	31	No
Pelican & Lice (ML) [8]	PML	35	35	Yes
Pelican & Lice (MP) [8]	PMP	35	35	Yes
Rodents & Pinworms [8]	RP	25	25	No
Primates & Pinworms [8]	PP	71	81	No
COG2085 [8]	COG2085	199	87	No
COG4965 [8]	COG4965	199	59	No
COG3715 [8]	COG3715	199	79	No
COG4964 [8]	COG4964	199	53	No

Since reconciliations obtained from planar co-phylogenetic trees are always planar, we restricted our experiments to non-planar instances. In order to obtain a datasuite of reconciliations we used the Eucalypt tool [8] to produce the set of minimum-cost reconciliations of each instance with costs 0, 2, 1, and 3 for co-speciation, duplication, loss, and host-switch, respectively. We configured the tool to filter out all time-inconsistent reconciliations based on the algorithm in [19]. Also, we bounded to 100 the reconciliations of each instance.

We implemented the two heuristics `SearchMaximalPlanar` and `Shorten-HostSwitch` in JavaScript (but we used Python for accessing the file system and the GDToolkit library [7] for testing planarity) and run the experiments on a Linux laptop with 7.7 GiB RAM and quadcore i5-4210U 1.70 GHz processor.

Table 2 shows the results of the experiments. Planar instances SB, EC, PML, and PMP were not used to generate reconciliations. Also, instances COG3715 and COG3715 did not produce any time-consistent reconciliation. For all the other phylogenetic-trees, the second column of Table 2 shows the number of reconciliations computed by Eucalypt (we bounded to 100 the reconciliations of RH, COG2085, and COG4965). Table 2 is vertically divided into three sections, each

Table 2. The results of the experiments.

Inst.	#Rec.	ShortenHostSwitch				SearchMaximalPlanar*				SearchMaximalPlanar			
		#Crossings			Avg ms	#Crossings			Avg ms	#Crossings			Avg ms
		Max	Min	Avg		Max	Min	Avg		Max	Min	Avg	
CM	64	30	15	21	0.5	21	13	17	644	20	10	16	485
FD	80	84	55	69	1	108	74	92	7289	110	67	91	4596
GL	2	1	1	1	0	1	1	1	180	2	2	2	67
PP	72	6	2	3	1	4	3	3	4840	2	1	1	1154
RH	100	11	11	11	2	11	9	10	1710	15	10	12	1701
RP	3	4	2	3	1	3	3	3	737	3	3	3	195
SC	1	6	6	6	0	4	4	4	499	4	4	4	166
SFC	16	22	11	17	0	16	12	13	412	20	11	15	355
COG2085	100	80	58	70	7	95	84	89	20540	99	68	82	17270
COG4965	100	125	79	97	8	68	57	60	9901	65	52	58	5636

devoted to a different heuristics. Each section shows the minimum, maximum, and average number of crossings and the average computation time for the HP-drawings produced by the heuristics on the reconciliations obtained for the phylogenetic-tree specified in the first column.

The section labeled **SearchMaximalPlanar*** shows the results of **SearchMaximalPlanar** where we computed the embedding of tree H (which is the most expensive algorithmic step) once for all the reconciliations of the same instance. Hence, differently from the other two sections, the computation times reported in this section refer to the sum of computation times for all the reconciliations obtained from the same instance.

From Table 2 it appears that heuristic **SearchMaximalPlanar** is much slower than **ShortenHostSwitch**. This could have been predicted, since **SearchMaximalPlanar** runs a planarity test several times. However, the gain in terms of crossings is questionable. Although there are instances where **SearchMaximalPlanar** appears to outperform **ShortenHostSwitch** (for example, CM, PP) this is hardly a general trend. We conclude that aiming at planarity is not the right strategy for minimizing crossings in this particular application context.

The strategy of computing the embedding of H once for all reconciliations of the same co-phylogenetic tree (central section labeled **SearchMaximalPlanar*** of Table 2) seems to be extremely effective in reducing computation times. For example, on instance **COG2085**, where this heuristics needed 20.5 s, it actually used 205 msec per reconciliation, about 11% of the time needed by **SearchMaximalPlanar**, at the cost of very few additional crossings.

8 Conclusions and Future Work

This paper introduces a new and intriguing simultaneous visualization problem, i.e. producing readable drawings of the reconciliations of co-phylogenetic trees.

Also, a new metaphor is proposed that takes advantage both of the space-filling and of the node-link visualization paradigms. We believe that such a hybrid strategy could be effective for the simultaneous visualization needs of several application domains.

As future work, we would like to address the problem of visually exploring and analyzing sets of reconciliations of the same co-phylogenetic tree, which is precisely the task that several researchers in the biological field need to perform. Heuristic `SearchMaximalPlanar*` is a first step in this direction, since it maintains the mental map of the user by fixing the drawing of H . Finally, we would like to adapt heuristics for the reduction of the crossings of tanglegram drawings, such as those in [3, 12, 16], to our problem and we would like to perform user tests to assess the effectiveness of the proposed metaphor.

Acknowledgments. We thank Riccardo Paparozzi for first experiments on the visualization of co-phylogenetic trees. Moreover, we are grateful to Marie-France Sagot and Blerina Sinimeri for proposing us the problem and for the interesting discussions.

References

1. CophyTrees - viewer associated with [8]. <http://eucalypt.gforge.inria.fr/viewer.html>
2. Bansal, M.S., Chang, W.-C., Eulenstein, O., Fernández-Baca, D.: Generalized binary tanglegrams: algorithms and applications. In: Rajasekaran, S. (ed.) BICoB 2009. LNCS, vol. 5462, pp. 114–125. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00727-9_13
3. Böcker, S., Hüffner, F., Truss, A., Wahlström, M.: A faster fixed-parameter approach to drawing binary tanglegrams. In: Chen, J., Fomin, F.V. (eds.) IWPEC 2009. LNCS, vol. 5917, pp. 38–49. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-11269-0_3
4. Buchin, K., Buchin, M., Byrka, J., Nöllenburg, M., Okamoto, Y., Silveira, R.I., Wolff, A.: Drawing (complete) binary tanglegrams. *Algorithmica* **62**(1–2), 309–332 (2012)
5. Calamoneri, T., Di Donato, V., Mariottini, D., Patrignani, M.: Visualizing co-phylogenetic reconciliations. Technical report [arXiv:1708.09691](https://arxiv.org/abs/1708.09691), Cornell University (2017)
6. Chevenet, F., Doyon, J.P., Scornavacca, C., Jacox, E., Jousset, E., Berry, V.: SylvX: a viewer for phylogenetic tree reconciliations. *Bioinformatics* **32**(4), 608–610 (2016)
7. Di Battista, G., Didimo, W.: Gdtoolkit. In: Tamassia, R. (ed.) *Handbook on Graph Drawing and Visualization*, pp. 571–597. Chapman and Hall/CRC, Boca Raton (2013)
8. Donati, B., Baudet, C., Sinimeri, B., Crescenzi, P., Sagot, M.F.: EUCALYPT: efficient tree reconciliation enumerator. *Algorithms Mol. Biol.* **10**(1), 3 (2015)
9. Dwyer, T., Schreiber, F.: Optimal leaf ordering for two and a half dimensional phylogenetic tree visualisation. In: *Proceedings of the 2004 Australasian Symposium on Information Visualisation - Volume 35, APVis 2004*, pp. 109–115. Australian Computer Society Inc., Darlinghurst (2004)

10. Fernau, H., Kaufmann, M., Poths, M.: Comparing trees via crossing minimization. *J. Comput. Syst. Sci.* **76**(7), 593–608 (2010)
11. Kruskal, J.B., Landwehr, J.M.: Icicle plots: better displays for hierarchical clustering. *Am. Stat.* **37**(2), 162–168 (1983)
12. Nöllenburg, M., Völker, M., Wolff, A., Holten, D.: Drawing binary tanglegrams: an experimental evaluation. In: Finocchi, I., Hershberger, J. (eds.) *ALENEX 2009*. pp. 106–119. SIAM (2009)
13. Ovia, Y., Fielder, D., Conow, C., Libeskind-Hadas, R.: The co phylogeny reconstruction problem is NP-complete. *J. Comput. Biol.* **18**(1), 59–65 (2011)
14. Rusu, A.: Tree drawing algorithms. In: Tamassia, R. (ed.) *Handbook on Graph Drawing and Visualization*, pp. 155–192. Chapman and Hall/CRC, Boca Raton (2013)
15. Schulz, H.J.: Treevis.net: a tree visualization reference. *IEEE Comput. Graphics Appl.* **31**(6), 11–15 (2011)
16. Scornavacca, C., Zickmann, F., Huson, D.H.: Tanglegrams for rooted phylogenetic trees and networks. *Bioinformatics* **13**(27), i248–i256 (2011)
17. Sennblad, B., Schreil, E., Sonnhammer, A.C.B., Lagergren, J., Arvestad, L.: primetv: a viewer for reconciled trees. *BMC Bioinform.* **8**(1), 148 (2007)
18. Stolzer, M., Lai, H., Xu, M., Sathaye, D., Vernot, B., Durand, D.: Inferring duplications, losses, transfers and incomplete lineage sorting with nonbinary species trees. *Bioinformatics* **28**(18), i409–i415 (2012)
19. Tofigh, A., Hallett, M.T., Lagergren, J.: Simultaneous identification of duplications and lateral gene transfers. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **8**(2), 517–535 (2011)
20. Tollis, I.G., Kakoulis, K.G.: Algorithms for visualizing phylogenetic networks. In: Hu, Y., Nöllenburg, M. (eds.) *GD 2016*. LNCS, vol. 9801, pp. 183–195. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_15
21. Wieseke, N., Hartmann, T., Bernt, M., Middendorf, M.: Cophylogenetic reconciliation with ILP. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **12**(6), 1227–1235 (2015)

Graph Layout Designs

Anisotropic Radial Layout for Visualizing Centrality and Structure in Graphs

Mukund Raj^(✉) and Ross T. Whitaker

University of Utah, Salt Lake City, USA
{mraj, whitaker}@cs.utah.edu

Abstract. This paper presents a novel method for layout of undirected graphs, where nodes (vertices) are constrained to lie on a set of nested, simple, closed curves. Such a layout is useful to simultaneously display the structural centrality and vertex distance information for graphs in many domains, including social networks. Closed curves are a more general constraint than the previously proposed circles, and afford our method more flexibility to preserve vertex relationships compared to existing radial layout methods. The proposed approach modifies the multidimensional scaling (MDS) stress to include the estimation of a vertex depth or centrality field as well as a term that penalizes discord between structural centrality of vertices and their alignment with this carefully estimated field. We also propose a visualization strategy for the proposed layout and demonstrate its effectiveness using three social network datasets.

Keywords: Centrality · Graph layout · Network visualization

1 Introduction

Graphs are an important data structure that are used to represent relationships between entities in a wide range of domains. An interesting aspect in graph analysis is the notion of (structural) *centrality*, which pertains to quantifying importance of entities (or vertices, nodes) within the context of the graph structure as defined by its relationships (or edges). The need to compute centrality and convey it through visualization is seen in many areas, for example, in biology [27], transportation [6] and social sciences [5]. In this work, we propose a method to visualize node centrality information in the context of overall graph structure, which we capture through intervertex (graph theoretical) distances. The proposed method determines a *layout* (positions of nodes on a 2D drawing) that meet the following two, often competing, criteria:

- *Preservation of distances:* The Euclidean (geometrical) distances in the layout should approximate, to the extent possible, the graph theoretical distances between the respective nodes.
- *Anisotropic radial monotonicity:* Along any ray traveling away from the position of the most central node, nodes with a lower centrality should be placed geometrically further along the ray.

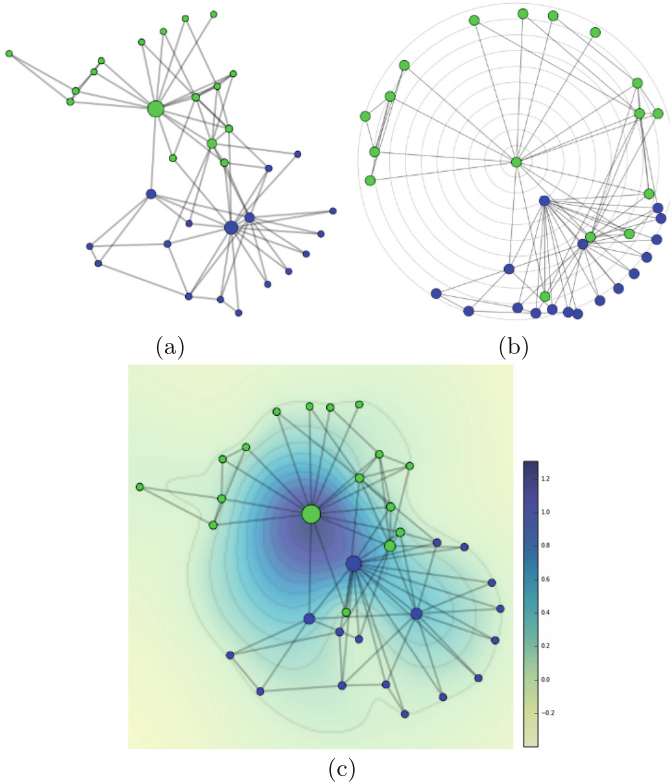


Fig. 1. Visualization of Zachary’s karate club social network using (a) MDS, (b) radial layout, and (c) anisotropic radial layout. Node sizes encode betweenness centrality.

We also introduce a visualization strategy for the proposed layout that further highlights the centrality and structure in the graph by using additional encoding channels, and demonstrate the benefits of our approach with real datasets (see Fig. 1 as an example).

Visualization methods for gaining insights from graph structured data are an important and active area of research. Significant efforts in this area are targeted toward developing effective layouts. Layout methods can have various goals that range from trying to reduce clutter and edge crossings [7] to faithfully representing the structure by preserving the distances between nodes and topological features [15]. As positions are the best way to graphically convey numbers [8], layouts are also used to convey numerically encoded measures of hierarchy or importance associated with nodes [5, 11].

Radial layouts have been shown to be an effective method to visually convey the relative *importance* of nodes, where importance may be defined, for instance, by a node’s *centrality* [5]. The centrality of a node is a quantification of its importance in a graph by considering its various structural properties, such

as, connectedness, closeness to others, and role as an intermediary [14, 34]. In conventional radial layouts, the distance of nodes from the geometric center (origin) of the layout depends *only* on the node’s centrality, and nodes with a higher centrality value are placed closer to the origin in the layout, often times forming rings or concentric circles.

Given a graph and centrality values associated with its nodes, several approaches have been proposed to determine a radial layout. One line of work, which deals with discrete centrality values, attempts to minimize edge crossings [1]. Another approach, which also tackles continuous centrality values, involves optimizing a *stress* energy (Sect. 2.2) by including a penalty for representation error (of graph distances) as well as deviation from radial constraints [5, 6]. The penalty acts a *soft* constraint wherein the solution is allowed to deviate from the constraint at the expense of increased local stress. The literature shows that radial constraints may also be included as a *hard* constraint by only allowing those solutions that satisfy the constraints [2, 12, 13].

While state-of-the-art methods for radial graph layout do effectively convey node centrality, the associate circular centrality constraints make it difficult to preserve other important, structural graph characteristics such as distances, which, in turn, makes it difficult to preserve the holistic structure of the graph. On the other hand, despite being effective in preserving the overall structure, general layout methods such as multidimensional scaling are often fail to readily convey centrality (e.g. by failing to ensure that structurally central nodes in the graph-theoretical sense appear near the center of the layout and vice versa). In this manuscript, we propose a method that simultaneously tackles both the above issues.

The underlying idea for the proposed layout algorithm is that we can relax the constraint that requires nodes with similar centrality to lie on a circle, and instead, allow for such nodes to be constrained by a more general shape: a simple closed curve or *centrality contour*. Centrality contours are nested isolevel curves on a smooth, radially decreasing estimate of node centrality values over a 2D field. We demonstrate that the additional flexibility in placing the nodes afforded by the centrality contours over circles, in conjunction with some additional visual cues in the background, lets us achieve a better trade off than existing methods in conveying centrality and general structure together.

2 Background

In this section, we describe the various underlying technicalities that are relevant to the proposed method, and begin with some notation/definitions.

We define a weighted, undirected graph $G(V, E, W)$ as a set of vertices (or nodes) V , a set of edges $E \subseteq V \times V$ and a set of edge weights, $W : E \mapsto \mathbb{R}^+$, assigned to each edge. We define n to be cardinality of node set; i.e., $n = |V|$. The graph-theoretical distance (shortest-path along edges) between two nodes u and v is denoted by d_{uv} . We denote a general position in a 2D layout as $\bar{x} = (x, y)$ and the Euclidean distance between two nodes u and v as $\delta(\bar{x}_i, \bar{x}_j) = \|\bar{x}_u - \bar{y}_v\|_2$.

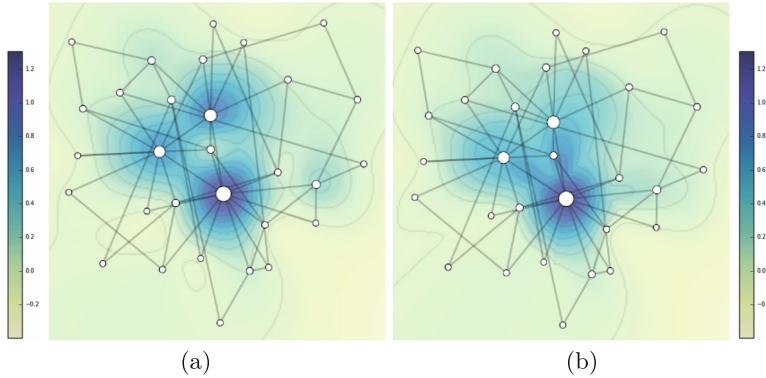


Fig. 2. An (a) *interpolation field* for node centrality values, and (b) the associated (radially) *monotonic field* for a 30 node random graph generated using the Barabasi-Albert model. Node positions are determined using MDS and node sizes encode betweenness centrality.

2.1 Centrality and Depth

The need to measure, and quantify, the importance of individual entities within the context of a group occurs in many domains. In graph analytics, this need is addressed by *centrality* indices, which are typically real-valued functions over the nodes of a graph [34]. The specific properties that qualify the importance of nodes may depend on the application or data type, and several methods to compute centrality have been proposed, such as degree centrality [14], closeness centrality [26], and betweenness centrality [14]. While the emphasis of the various centrality definitions can be different, they all share a common characteristic of depending only on the *structure* of the graph rather than parameters associated with the nodes [34]. For the examples in this paper we use betweenness centrality due to its relevance to the datasets (Sect. 4).

The *betweenness centrality* of a node, $v \in G$, is defined as the percentage (or number) of shortest paths in the entire graph G that pass through the node v . As shown in work of Raj et al. [23], barring instances of multiple geodesics, betweenness centrality is a special case of a more general notion of *vertex depth* on graphs—a generalization of data depth to vertices on graphs. Data depth is a family of methods from descriptive statistics that attempts to quantify the idea of centrality for ensemble data without any assumption of the underlying distribution. Data depth methods often rely on the formation of *bands* from convex sets and the probability of a point lying within a randomly chosen band. The extension of band depth to graphs [23] relies on the convex closure of a set of points (via shortest paths), and thereby generalizes betweenness centrality by considering bands formed by sets of nodes, rather than only the shortest paths between pairs of nodes, and allows for a nonuniform probability distribution over the nodes of the graph.

In addition to graphs, data depth methods have been proposed for several other data types such as points in Euclidean space [29], functions [19], and curves [20, 22]. Despite their distinct formulations, data depth methods are expected to share a few common desirable properties [33] such as: 1. maximum at geometric center 2. zero at infinity 3. radial monotonicity; which make data depth an attractive basis for ensemble visualization methods [22, 25, 28]. Graph centrality is a type of data depth on the nodes of a graph, and here we pursue layout methods that convey these depth properties.

2.2 Stress and Multidimensional Scaling (MDS)

Our proposed method is based on a modification to the MDS objective function, and therefore we give a brief summary of MDS. MDS is family of methods that help visualize the similarity (or dissimilarity) between members in a data set [4]. Over the years, MDS has been the foundation for a range of graph drawing algorithms that aim to achieve an isometry between graph theoretical and Euclidian distances between nodes [6, 17]. From among various types of MDS methods that exist, here we consider *metric MDS with distance scaling*, which is popular in the graph drawing literature [15] (see Fig. 2 for an example).

In the context of graph drawing, given a distance matrix based on graph-theoretical distance, the goal is to find node positions $X = \{\bar{x}_i : 1 \leq i \leq n\}$ that minimize the following sum of squared residuals—also known as *stress*:

$$\sigma(X) = \sum_{u,v} w_{uv} (d_{uv} - \|\bar{x}_u - \bar{x}_v\|_2)^2, \tag{1}$$

where $w_{uv} \geq 0$ is the weighting term for residual associated with pair u, v . In the proposed work we employ a standard weighting scheme for graphs, known as *elastic scaling* [21], by setting $w_{uv} = d_{uv}^{-2}$. This gives preference to local distances by minimizing *relative* error rather than *absolute* error during the optimization.

Node positions that minimize the objective (Eq. (1)) have been shown to be visually pleasing and convey general structure of the graph [17]. Although, the state-of-the-art approach for optimizing the objective function is *stress majorization* [15], we employ standard *gradient descent* because of its compatibility with the proposed modification to the objective (Sect. 3). The gradient of the standard MDS objective is as follows [4]:

$$\nabla \sigma(X) = 2VX - B(X)X \tag{2}$$

where matrices $V = (v_{ij})$ and $B = (b_{ij})$, with $1 \leq i, j \leq n$, can be compactly represented as:

$$v_{ij} = \begin{cases} -w_{ij} & \text{for } i \neq j \\ \sum_{j=1, j \neq i}^n w_{ij} & \text{for } i = j \end{cases} \quad b_{ij} = \begin{cases} -\frac{w_{ij}d_{ij}}{\delta(\bar{x}_i, \bar{x}_j)} & \text{for } i \neq j \text{ and } \delta(\bar{x}_i, \bar{x}_j) \neq 0 \\ 0 & \text{for } i \neq j \text{ and } \delta(\bar{x}_i, \bar{x}_j) = 0 \end{cases}$$

$$b_{ii} = - \sum_{j=1, j \neq i}^n b_{ij}.$$

2.3 Strictly Monotone and Smooth Regression

The proposed method also relies on the construction of a smooth and radially decreasing approximation of centrality values over a 2D field, which we call the *monotonic field* (Fig. 2). The first part of this construction is an interpolation of centrality values of sparsely located nodes on the layout to obtain a dense 2D field, which we call the *interpolation field* (Fig. 2a). For this we use *thin plate splines* [3] interpolation, a standard technique for interpolating unstructured data which produces optimally smooth fields.

The next part is to construct a radially monotonic approximation of the interpolation field. We devote the rest of this section to a brief description of the method that we use for constructing this approximation (monotonic field), which is adapted from Dette et al. [9, 10].

For a 1D function [9], $m(t) : [0, 1] \rightarrow \mathbb{R}$, an elegant algorithm for computing its monotonic approximation $\hat{m}_A(t)$ proceeds as follows in *two* steps [9]:

- *Step 1 (Monotonization)*: Construct a density estimate from sampled values of input function m and use it as input to compute an estimate of the inverse of the regression function \hat{m}_A^{-1} .

$$\hat{m}_A^{-1}(t) = \frac{1}{Q\omega} \sum_{i=1}^Q \int_{-\infty}^t K\left(\frac{m(\frac{i}{Q}) - u}{\omega}\right) du, \tag{3}$$

where Q is the parameter controlling the sampling density, K is a continuously differentiable and symmetric kernel, and ω is the bandwidth. Here, \hat{m}_A^{-1} is a strictly *increasing* estimate of m^{-1} , however, we can easily obtain a strictly *decreasing* estimate by reversing the limits on the integral in Eq. (3).

- *Step 2 (Inversion)*: Obtain the final estimate of \hat{m}_A by numerically inverting \hat{m}_A^{-1} .

In order to obtain an approximation to a 2D function that is monotonic along radial lines emanating from the deepest or most central node, we use a polar coordinate representation of the field. We build the polar representation by sampling the interpolation field along 360 evenly spaced, center outward rays. The idea is to repeatedly monotonize the interpolation field with respect to a single variable i.e., for a fixed value of the angular coordinate, obtain a (1D) estimate that is strictly decreasing along the radial coordinate. We then repeat this process, successively monotonizing 1D functions that correspond to each value of angular coordinate in its (discrete) domain; see Fig. 2b for an example of the resulting monotonic field. The spline interpolation is smooth, and by the properties of the monotonic approximation (see [10]), the resulting monotonic field is smooth (except at origin, where polar the coordinates maybe nonsmooth).

3 Method

Here we describe our method in two parts. First is the layout algorithm (Sect. 3.1), and second is a visualization strategy (Sect. 3.2) that complements the layout to simultaneously convey graph structure and node centrality.

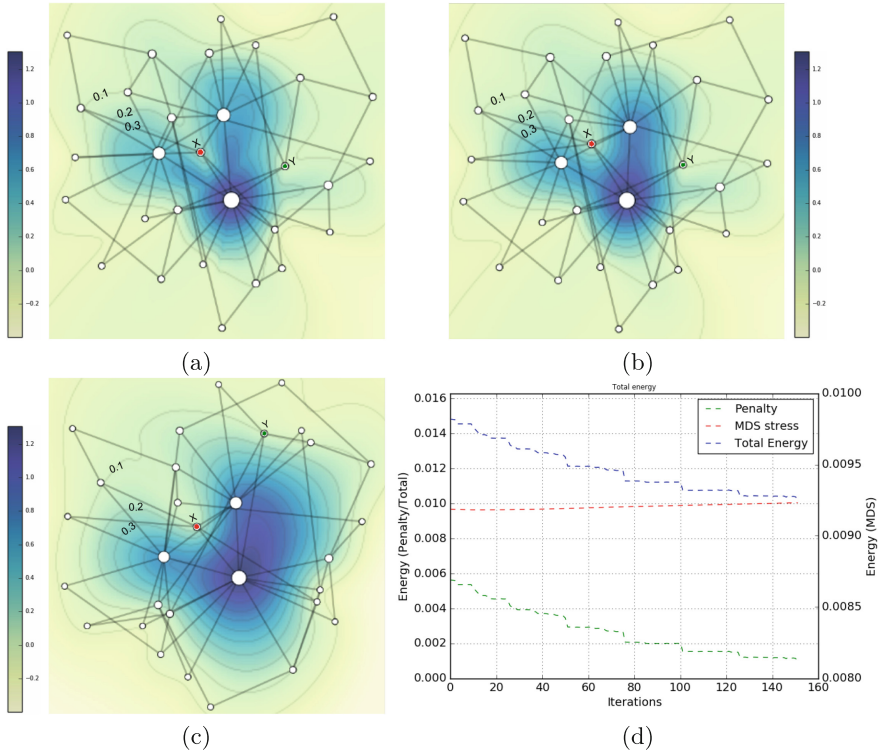


Fig. 3. Sensitivity of anisotropic radial layout to penalty weights for the graph in Fig. 2: (a) $w_\rho = 0.1$, (b) $w_\rho = 1$, (c) $w_\rho = 10$; centrality contours with isovalues 0.1, 0.2 and 0.3 as well as nodes X (red) and Y (green) with centrality values 0.2 and 0.1 are identified, and (d) a typical plot of objective energy during the optimization process ($w_\rho = 1$). (Color figure online)

3.1 Anisotropic Radial Layout

In addition to preserving the graph-theoretical distances, we also aim to place every node on a radially monotonic approximation of a centrality field—called the *monotonic field* (Sect. 2.3)—such that the value of the field at the location of the node is equal to the centrality value of the node. We accomplish this by modifying the (distance preserving) MDS objective or *stress* (Sect. 2.2) to incorporate the following penalty term, which penalizes the deviation of monotonic field values from the node centrality values:

$$\rho(X) = (M_{X,\bar{c}}(X) - \bar{c})^2 \quad (4)$$

where $\bar{c} \in \mathbb{R}^n$ is a vector of node centrality values and $X \in \mathbb{R}^{n \times 2} = \{\bar{x}_i : 1 \leq i \leq n\}$ denotes associated node positions. $M_{X,\bar{c}}(X) \in \mathbb{R}^n$ denotes a vector of values of the 2D monotonic field at locations X . The symbols in the subscript (X and \bar{c}) denote the use of node positions and centrality values in the

construction of the monotonic field. In the limiting case where the *interpolation field* (Sect. 2.3) itself is monotonic, the value of this penalty term drops to zero. Our final objective is a sum of the MDS stress and the above penalty term, and can be stated as follows:

$$\gamma(X) = \underbrace{\sigma(X)}_{\text{MDS stress}} + w_\rho \rho(X) \quad (5)$$

where w_ρ is a weighting factor that controls the influence of the penalty, with respect to the MDS stress. The gradient of the modified objective above is obtained as:

$$\nabla\gamma(X) = \nabla\sigma(X) + w_\rho \times \underbrace{2(M_{X,\bar{c}}(X) - \bar{c}) \odot \nabla M_{X,\bar{c}}(X)}_{\nabla\rho(X)}, \quad (6)$$

where \odot denotes element wise product. It is difficult to compute the gradient of $M_{X,\bar{c}}(X)$ because of dependence of M on X and the associated process for monotonic approximation. Therefore, we let the field lag, and treat X (in subscript) as a constant when numerically approximating the gradient of M . We deal with the resulting accumulation of error by recomputing the depth field after a fixed number of iterations, or *lag*, denoted by ℓ .

The parameters w_ρ and ℓ need to be chosen carefully. w_ρ needs to be set to find a balance between preserving the intrinsic graph structure and ensuring that the centrality of nodes match the field value at their position. Figure 3a-c show, respectively, results of a *small* w_ρ unable to move nodes to appropriate positions with regard to the field (observe nodes X, Y), an *intermediate* w_ρ , and a *large* w_ρ resulting in unnecessary structural distortion with regard to initial positions (observe node Y). The parameter ℓ controls the lag of the monotonic field; if ℓ is too small, the frequent updates can lead to instabilities, while values that are too large can cause slow convergence. A typical energy profile during optimization is shown in Fig. 3d; where the sharp changes in the total energy correspond to the updates of the monotonic field. We encourage the layout to be as similar as possible to the MDS layout by initializing the node positions as determined by an *unmodified* MDS objective [15]. The entire process, as summarized in Algorithm 1, iterates until updates no longer result in significant changes to node positions.

The computational complexity of a single iteration is $\mathcal{O}(n^3)$ due to the step of computing the monotonic field which involves interpolation using thin plate spline. However, we only update the field once every ℓ iterations. This leads to a complexity of $\mathcal{O}(n^2)$ (same as MDS) for a large majority of iterations.

3.2 Visualization

In this layout, nodes are constrained to lie on level sets of centrality, which are *general* closed curves, rather than circles, and the shapes of these curves depend on the structure of the graph. Therefore, we can help interpretability of the layout and reduce cognitive load for the user by providing additional cues for shapes

Algorithm 1. Layout with anisotropic radial constraints

Input: Graph $G = \{V, E, W\}$, maximum number of iterations $k \in \mathbb{N}$, depth field lag ℓ , step size α , weighing factor w_p

Output: Positions $X = \{\bar{x}_i : 1 \leq i \leq n\}$ for all $v_i \in V$

```

 $n \leftarrow |V|$ 
 $X_0 \leftarrow$  initialize node positions using MDS ; /* (Sect. 2.2) */
 $\bar{c} \in \mathbb{R}^n \leftarrow$  compute graph centrality values for  $v_i \in V$ 
 $j \leftarrow -1$  ; /* index to keep track of field updates */
for  $t = 1, \dots, k$  do
  if  $t \bmod \ell = 0$  then
     $j \leftarrow j + 1$ 
     $X_j \leftarrow X_t$ 
     $M_{X_j, \bar{c}}(X_t) \leftarrow$  compute monotonic field ; /* (Sect. 2.3) */
  end
   $X_{t+1} \leftarrow X_t - \alpha \left( \nabla \sigma(X_t) + w_p \times 2(M_{X_j, \bar{c}}(X_t) - \bar{c}) \odot \nabla M_{X_j, \bar{c}}(X_t) \right)$ ;
  /* gradient update step (Sect. 3.1) */
end

```

of these curves. We provide cues in the form of faded renderings of centrality contours (isolines on the monotonic field) and a monotonic field colormap in the background. The radial monotonicity described in Sect. 3.1 ensures that the contours are nested curves that enclose a *common* maxima (at origin); leading to a bijective mapping between contours and centrality values, and pushing nodes to lie on the *unique* contour that corresponds to their centrality. In this paper, we normalize node centrality to fall between 0 and 1; and show 10 contour curves that evenly span this range. We also use node size as an extra encoding channel for centrality—in addition to location—to further highlight the centrality structure. We can, of course, use the size channel to encode centrality even with the standard MDS layout, however, that approach can lead to the issue of conflicting centrality cues from size and location channels (see image (a) in Figs. 1, 4 and 5).

4 Results

4.1 Zachary’s Karate Club

The Zachary’s karate club graph is a well known data set that is a social network of friendships in a karate club at a US university, as recorded during a study [32]. This graph contains 34 nodes, each representing an individual, and 78 unweighted edges that represent a friendship between the associated individuals (Fig. 1). During the period of observation, a conflict between two key members, identified as the “administrator” and “instructor”, leads to a split in the club, giving it an interesting two cluster structure. In Fig. 1, nodes representing members who are part of the instructor’s and administrator’s groups are drawn in green and blue, respectively.

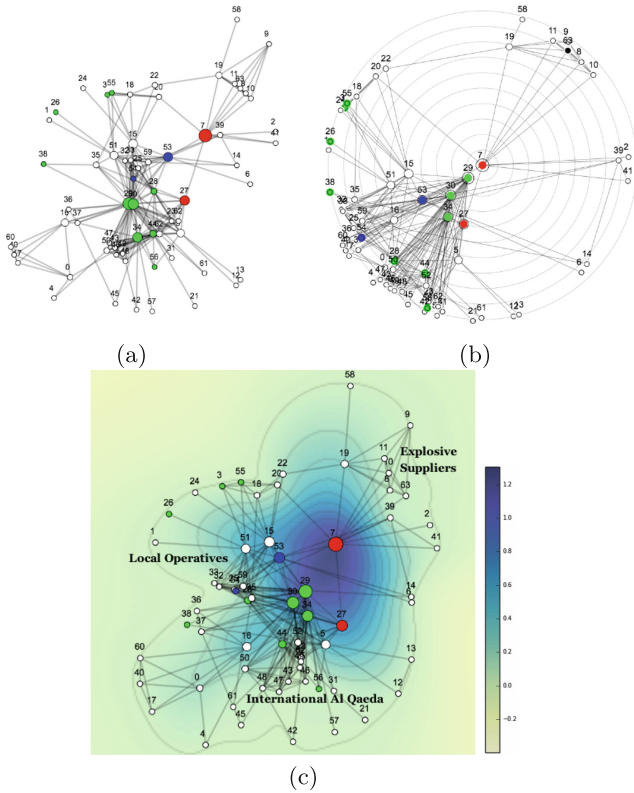


Fig. 4. Network of terrorists and affiliates connected to the 2004 Madrid train bombing using (a) MDS, (b) radial layout, (c) anisotropic radial layout. (Color figure online)

Figure 1 shows three different visualizations of the karate club network: MDS, radial layout (from [6]), and anisotropic radial layout (ARL). We can make a few observations from the visualizations. While MDS does a good job of preserving the two clusters, it does not unambiguously convey centrality. On the other hand, radial layout clearly showcases the centrality at the expense of dispersing the clusters by distorting distances among their nodes, thereby obscuring their internal structure. We see that ARL is able to largely preserve the structure seen in MDS with clearly distinguishable clusters, and also clearly convey the centrality information. While radial layout pushes the instructor’s group far away due to low betweenness centrality, ARL lets them remain close by *bringing in* the outermost contour toward to the group instead. Similarly, the administrator is also allowed to remain closer to their group by the protrusion of the inner contours, which enclose the most central nodes, toward the administrator.

4.2 Terrorist Network from 2004 Madrid Train Bombing

Figure 4 shows visualizations of a network of individuals connected to the bombing of trains in Madrid on March 11, 2004. This data was originally compiled by

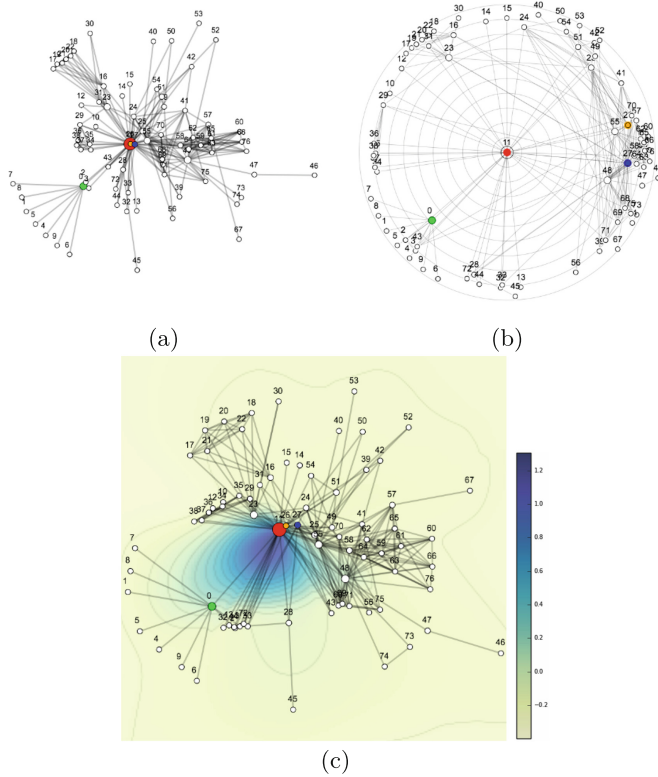


Fig. 5. Coappearance network for characters in the novel *Les Miserables* using (a) MDS, (b) radial layout, (c) anisotropic radial layout. (Color figure online)

Rodriguez [24] from newspaper articles that reported on the subsequent police investigation. There are 64 nodes that represent suspects and their relatives, and 243 edges that have weights ranging from 1 to 4 which represent an aggregated strength of connection based on various parameters such as contact, kinship, ties to Al Qaeda, etc [16]. In Fig. 4, (as well as Fig. 5), distances between nodes are related inversely to edge weights. In the visualization, we identify nodes using numbers to avoid text clutter, however, we include a mapping to names of individuals represented by the nodes in the Appendix.

Rodriguez [24] identifies several key suspects as follows: ring leaders (marked in blue in Fig. 4), members of a field operating group who were closely involved with the actual carrying out the attack (green), intermediaries (red), as well as suspects with local roots, ties to foreign Al Qaeda, and those who supplied explosives. On comparing the visualizations in Fig. 4 we see that ARL (Fig. 4c) is able to better preserve the structure and cohesiveness of the core members of the field operating group in comparison to the radial layout (Fig. 4b). Critically, a key mastermind in this event, despite having a low centrality (due to communicating

often through an intermediary), is allowed to be close to the center in the ARL. This arrangement, possible due to the ability of centrality contours to adapt to the circumstance, preserves the close association between the masterminds that is lost in the radial layout. We also see that the flexibility of contours in ARL preserves the locality of various groups, which allows us to see the role of intermediaries with high centrality in acting as a bridge between various groups.

4.3 Coappearance Network for Characters in *Les Miserables*

The third dataset is a graph of character associations in the famous French novel *Les Miserables* (Fig. 5) [18]. This graph consists of 77 nodes, each representing a character in the novel, and 254 weighted edges where the weights represent the number of chapters that feature both characters associated with an edge.

We see that the main protagonist *Valjean* (marked in red) is placed prominently in all three visualizations (Fig. 5). However, other key characters in the plot such as *Inspector Javert* (blue) and *Cosett* (orange), who do not appear often with characters other than the protagonist (and thus have low betweenness centrality) are treated differently. While the radial layout relegates them to the periphery (far from Valjean) (Fig. 5b), MDS (Fig. 5a) paints a conflicting picture with regard to their centrality, e.g., Cosett's node almost overlaps with Valjean despite its low centrality. In contrast, the proposed ARL (Fig. 5c) is able to coherently convey the low centrality of the Inspector Javert and Cosett, as well as, their closeness to Valjean. The above issue of distance distortion appears to be a frequent occurrence in the radial layout due to many characters who have a low centrality value causing them to end up being packed in the outer periphery. A case of contrast is that of the character *Bishop Myriel* (green) who despite being associated with several characters, is only seen with Valjean once.

5 Discussion

This paper describes an energy-based layout algorithm for graphs, called *anisotropic radial layout*, which conveys structural centrality using *anisotropic*, radial constraints, while also preserving approximate distances (or structure) in the graph. In contrast to existing methods for conveying node centrality which employ an *isotropic* centrality field [2, 6], the proposed method determines an *anisotropic* centrality field on which to project nodes. While the energy minimization strategy described in this paper allows the solution to deviate from constraints, one can enforce hard constraints by adding a post processing step that projects nodes onto the closest position on their associated isocontour.

The key implication of the anisotropic centrality field in our method is that more central nodes are allowed to be placed further from origin than less central nodes—without an energy penalty—if they do not lie on a common ray; which aids our objective of achieving a better balance between visual representations of centrality and structure than possible with existing methods. Our objective differs from other prior work that use centrality or continuous fields to visualize structure of dense graphs [30, 31].

References

1. Bachmaier, C.: A radial adaptation of the sugiyama framework for visualizing hierarchical information. *IEEE Trans. Visual Comput. Graphics* **13**(3), 583–594 (2007)
2. Baingana, B., Giannakis, G.B.: Embedding graphs under centrality constraints for network visualization. arXiv preprint [arXiv:1401.4408](https://arxiv.org/abs/1401.4408) (2014)
3. Bookstein, F.L.: Principal warps: thin-plate splines and the decomposition of deformations. *IEEE Trans. Pattern Anal. Mach. Intell.* **11**(6), 567–585 (1989)
4. Borg, I., Groenen, P.J.: *Modern Multidimensional Scaling: Theory and Applications*. Springer Science & Business Media, New York (2005). <https://doi.org/10.1007/0-387-28981-X>
5. Brandes, U., Kenis, P., Wagner, D.: Communicating centrality in policy network drawings. *IEEE Trans. Visual Comput. Graphics* **9**(2), 241–253 (2003)
6. Brandes, U., Pich, C.: More flexible radial layout. In: Eppstein, D., Gansner, E.R. (eds.) *GD 2009*. LNCS, vol. 5849, pp. 107–118. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11805-0_12
7. Brandes, U., Wagner, D.: Analysis and visualization of social networks. In: Jünger, M., Mutzel, P. (eds.) *Graph Drawing Software*, pp. 321–340. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-642-18638-7_15
8. Cleveland, W.S., McGill, R.: Graphical perception: theory, experimentation, and application to the development of graphical methods. *J. Am. Stat. Assoc.* **79**(387), 531–554 (1984)
9. Dette, H., Neumeyer, N., Pilz, K.F., et al.: A simple nonparametric estimator of a strictly monotone regression function. *Bernoulli* **12**(3), 469–490 (2006)
10. Dette, H., Scheder, R.: Strictly monotone and smooth nonparametric regression for two or more variables. *Can. J. Stat.* **34**(4), 535–561 (2006)
11. Dwyer, T., Koren, Y., Marriott, K.: IPSep-CoLa: an incremental procedure for separation constraint layout of graphs. *IEEE Trans. Visual Comput. Graphics* **12**(5), 821–828 (2006)
12. Dwyer, T.: Scalable, versatile and simple constrained graph layout. In: *Computer Graphics Forum*, vol. 28, pp. 991–998. Wiley Online Library (2009)
13. Dwyer, T., Koren, Y., Marriott, K.: Constrained graph layout by stress majorization and gradient projection. *Discrete Mathe.* **309**(7), 1895–1908 (2009)
14. Freeman, L.C.: Centrality in social networks conceptual clarification. *Soc. Netw.* **1**(3), 215–239 (1978)
15. Gansner, E.R., Koren, Y., North, S.: Graph drawing by stress majorization. In: Pach, J. (ed.) *GD 2004*. LNCS, vol. 3383, pp. 239–250. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31843-9_25
16. Hayes, B.: Connecting the dots can the tools of graph theory and social-network studies unravel the next big plot? *Am. Sci.* **94**(5), 400–404 (2006)
17. Kamada, T., Kawai, S.: An algorithm for drawing general undirected graphs. *Inf. Process. Lett.* **31**(1), 7–15 (1989)
18. Knuth, D.E.: *The Stanford GraphBase: a platform for combinatorial computing*, vol. 37 (1994)
19. López-Pintado, S., Romo, J.: On the concept of depth for functional data. *J. Am. Stat. Assoc.* **104**(486), 718–734 (2009)
20. López-Pintado, S., Sun, Y., Lin, J., Genton, M.: Simplicial band depth for multivariate functional data. *Adv. Data Anal. Classif.* **8**, 1–18 (2014)

21. McGee, V.E.: The multidimensional analysis of elastic distances. *Br. J. Math. Stat. Psychol.* **19**(2), 181–196 (1966)
22. Mirzargar, M., Whitaker, R., Kirby, R.: Curve boxplot: generalization of boxplot for ensembles of curves. *IEEE Trans. Visual Comput. Graphics* **20**(12), 2654–2663 (2014)
23. Raj, M., Mirzargar, M., Ricci, R., Kirby, R.M., Whitaker, R.T.: Path boxplots: a method for characterizing uncertainty in path ensembles on a graph. *J. Comput. Graph. Stat.* **26**(2), 243–252 (2017)
24. Rodríguez, J.A., Rodríguez, J.A.: The March 11th terrorist network: in its weakness lies its strength (2005)
25. Rousseeuw, P.J., Ruts, I., Tukey, J.W.: The bagplot: a bivariate boxplot. *Am. Stat.* **53**(4), 382–387 (1999)
26. Sabidussi, G.: The centrality index of a graph. *Psychometrika* **31**(4), 581–603 (1966)
27. Schreiber, F., Dwyer, T., Marriott, K., Wybrow, M.: A generic algorithm for layout of biological networks. *BMC Bioinf.* **10**(1), 375 (2009)
28. Sun, Y., Genton, M.G.: Functional boxplots. *J. Comput. Graph. Stat.* **20**(2), 316–334 (2011)
29. Tukey, J.W.: *Mathematics and the picturing of data* (1975)
30. Van Ham, F., Wattenberg, M.: Centrality based visualization of small world graphs. In: *Computer Graphics Forum*, vol. 27, pp. 975–982. Wiley Online Library (2008)
31. Van Liere, R., De Leeuw, W.: Graphsplatting: visualizing graphs as continuous fields. *IEEE Trans. Visual Comput. Graphics* **9**(2), 206–212 (2003)
32. Zachary, W.W.: An information flow model for conflict and fission in small groups. *J. Anthropol. Res.* **33**(4), 452–473 (1977)
33. Zuo, Y., Serfling, R.: General notions of statistical depth function. *Ann. Statist.* **28**, 461–482 (2000)
34. Zweig, K.A., et al.: Network analysis literacy. In: *MMB & DFT 2014*, p. 3 (2014)

Computing Storyline Visualizations with Few Block Crossings

Thomas C. van Dijk, Fabian Lipp^(✉), Peter Markfelder,
and Alexander Wolff^(iD)

Lehrstuhl für Informatik I, Universität Würzburg, Würzburg, Germany
fabian.lipp@uni-wuerzburg.de
<http://www1.informatik.uni-wuerzburg.de/en/staff/>

Abstract. Storyline visualizations show the structure of a story, by depicting the interactions of the characters over time. Each character is represented by an x -monotone curve from left to right, and a meeting is represented by having the curves of the participating characters run close together for some time. There have been various approaches to drawing storyline visualizations in an automated way. In order to keep the visual complexity low, rather than minimizing pairwise crossings of curves, we count *block crossings*, that is, pairs of intersecting bundles of lines.

Partly inspired by the ILP-based approach of Gronemann et al. [GD 2016] for minimizing the number of pairwise crossings, we model the problem as a satisfiability problem (since the straightforward ILP formulation becomes more complicated and harder to solve). Having restricted ourselves to a decision problem, we can apply powerful SAT solvers to find optimal drawings in reasonable time. We compare this SAT-based approach with two exact algorithms for block crossing minimization, using both the benchmark instances of Gronemann et al. and random instances. We show that the SAT approach is suitable for real-world instances and identify cases where the other algorithms are preferable.

1 Introduction

A storyline visualization is a particular abstraction of the structure of a narrative. A good visualization reveals the underlying structure by removing the details of how the story is presented and, instead, focusing on which entities interact as time passes within the narrative. This type of diagram was originally conceived to visualize meetings between characters in movies and, though it has since been interpreted more generally as an elegant way to visualize a sequence of interconnected interactions over time, the term storyline visualization remains.

In a storyline visualization, each character is represented by an x -monotone curve in the plane; we will refer to curves and characters interchangeably. Time goes from left to right, and a meeting between a set of characters (occurring for

The full version of this paper is available on arXiv [4].

F. Lipp was supported by Cusanuswerk.

the duration of a given time interval) is represented by a corresponding region in the plane where those curves come closely together. This drawing style is commonly attributed to Munroe [12], who represented several popular movies in this fashion. See Fig. 1 for an example drawn using our system.

Block Crossings in Storyline Visualization. When formalizing the drawing of storyline visualizations as an optimization problem, it is natural to minimize the number of crossings among the characters. As with graph drawing in general, this is not the be-all-end-all objective. For example, two groups of curves crossing each other in a grid structure are easier to understand visually than the same number of crossings scattered wildly throughout the drawing. In this paper we continue the study of such *block crossings* in storyline visualization.

Intuitively, a block crossing consists of two sets of locally parallel curves intersecting each other without any further curves in the crossing area [3]. (A formal definition is given below.) In the design of his movie narrative charts, Munroe seems aware (at least implicitly) of the concept of block crossings. Indeed, the Gestalt principle of “continuity” or “good continuation” [14] suggests that block crossings are easier to read, but what exactly makes the most readable drawing should be analyzed in proper user studies. Here we focus on practical computational aspects, having decided to minimize block crossings.

Concurrent Meetings. An important modeling decision that the literature has handled variously is whether it is possible for multiple meetings to occur at overlapping time intervals. Some papers define the input to the storyline visualization problem such that these *concurrent* meetings are impossible, for example by representing the meetings as a totally ordered set. Whether or not it is important to support concurrent meetings is open for discussion. One could, for example, represent each scene of a movie as a separate meeting that includes precisely the characters that participate: then meetings do not overlap. However, this is a rather mechanical interpretation of what storyline visualizations are for. Indeed, rather than strictly following the order of appearance in the movies, Munroe’s “movie narrative charts” [12] visualize the spatio-temporal structure underlying the story, rather than the presentation of the story: the x -axis in his charts represents time within the story, not time in the movie.¹ This paper supports concurrent meetings.

Previous Work. Tanahashi and Ma [13] computed storyline visualizations automatically and discuss various aesthetic criteria to be optimized. Kim et al. [9] used storylines to visualize genealogical data: meetings correspond to marriages and special techniques are used to indicate child–parent relationships.

¹ For example, Gandalf meets Éomer *while* the host of elves arrives at Helm’s Deep. In the movie, we learn about this only afterward; Munroe’s visualization of *The Lord of the Rings* [12] makes clear that this is concurrent.

Kostitsyna et al. [11] formalized the problem of crossing minimization for storylines. Their aim was to minimize the number of pairwise crossings (that is, *not* block crossings) in storylines. They proved the problem NP-hard, presented an FPT algorithm, and gave an upper bound on the number of crossings in a restricted setting. Gronemann et al. [8] designed an integer linear program (ILP) to minimize the number of pairwise crossings and evaluated it experimentally. Their approach is able to solve instances with 10–20 characters and up to about 50 meetings from real-world movies (and to a lesser degree, books) to optimality in a few seconds.

In an earlier paper [3], we introduced the concept of minimizing block crossings for drawing storylines. We showed that block crossing minimization in storylines is NP-hard. For special cases, we provided an approximation algorithm. Of particular relevance to the current paper are two exact algorithms, one of which is fixed-parameter tractable (FPT) in the number of characters.

The current paper improves on the above in two ways. Firstly, we have developed a new SAT-based algorithm for computing optimal storyline visualizations. We note that SAT formulations have been used before in graph drawing, for example by Bekos et al. [2]. Whereas we previously restricted ourselves to meetings that are points in time, we now handle concurrent meetings. (This more general problem is clearly NP-hard as well.) Secondly, we have now implemented the exact algorithms of our earlier paper [3], which enables an experimental evaluation and comparison. We see that the new algorithm is able to handle larger realistic instances than our previous algorithms, but that the FPT algorithm also has practical relevance.

Problem Definition. We generalize the problem statement compared to our previous paper [3] in order to handle the instances used by Gronemann et al. [8]. In this more general statement, we support meetings that span a certain amount of time (instead of allowing only instantaneous meetings); thus, meetings can overlap with other meetings. Additionally, we allow for birth and death of characters, that is, each character is only drawn in the storyline during its *lifespans* (being a set of time intervals).

A storyline \mathcal{S} is a triple (C, M, E) where $C = \{1, \dots, \kappa\}$ is a set of *characters*, $M = \{m_1, m_2, \dots, m_n\}$ is a set of meetings, and $E: C \rightarrow \mathcal{P}(\mathbb{IR})$ describes the lifespans of a character with \mathbb{IR} being the set of intervals of real numbers. A *meeting* m_j is a triple (s_j, e_j, C_j) where $s_j \in \mathbb{R}$ is the start time of the meeting, $e_j \in \mathbb{R}$ is the end time of the meeting, $s_j < e_j$, and $C_j \subseteq C$ contains the involved characters. A meeting m_j is said to be *active* at time $t \in \mathbb{R}$ if $t \in [s_j, e_j)$. The set $E(i) = \{[b_i^1, d_i^1], \dots, [b_i^{\eta_i}, d_i^{\eta_i}]\}$ contains the lifespans of character i , that is, η_i disjoint time intervals, in which the character is alive. For each of these time intervals $(1 \leq \iota \leq \eta_i)$, b_i^ι describes the “birth” while d_i^ι describes the “death” of the character. Character i is said to be *alive* at time $t \in \mathbb{R}$ if $t \in I$ for some $I \in E(i)$.

We forbid that a character participates in two meetings at the same time: in our drawing style, it wouldn’t be possible to distinguish the two groups. More formally, for any two meetings m_j, m_ℓ with $s_j < s_\ell < e_j$, we require

that $C_j \cap C_\ell = \emptyset$. Obviously, character i can only be part of a meeting m_j if i is alive during the time span of the meeting, that is, if $[s_j, e_j] \subseteq I$ for some $I \in E(i)$. In particular, a character cannot be born or die during a meeting.

A solution for a storyline instance $\mathcal{S} = (C, M, E)$ consists of a sequence $\Pi = [\pi_1, \dots, \pi_\lambda]$ of permutations of subsets of C and a nondecreasing function $A: \mathbb{R} \rightarrow \{1, \dots, \lambda\}$ describing the connection between points in time and the permutations in the solution. A solution is *admissible* if it fulfills the following conditions.

- (a) For any point in time $t \in \mathbb{R}$,
 - (i) $\pi_{A(t)}$ contains exactly the characters that are alive at time t , and
 - (ii) for any meeting that is active at time t , its set of characters must be a contiguous block in $\pi_{A(t)}$.
- (b) For $p \in \{2, \dots, \lambda\}$:
 - (i) If the character sets of π_{p-1} and π_p are identical, then either π_{p-1} and π_p are identical or they differ in a *block crossing*, that is, two adjacent blocks of characters switch their order. Suppose that, after renumbering, $\pi_{p-1} = \langle 1, \dots, a, \dots, b, \dots, c, \dots, \kappa \rangle$. Then exchanging the two adjacent blocks $\langle a, \dots, b \rangle$ and $\langle b+1, \dots, c \rangle$ yields the permutation $\pi_p = \langle 1, \dots, a-1, b+1, \dots, c, a, \dots, b, c+1, \dots, \kappa \rangle$.
 - (ii) If the character sets of π_{p-1} and π_p are not identical, then their intersection must be in the same order in π_{p-1} and in π_p . They need not remain contiguous.

Now we can formally state the problem that we consider in this paper, *General Storyline Block Crossing Minimization*: Given a storyline instance (C, M, E) , find an admissible solution (Π, A) that minimizes the number of block crossings.

We define \mathcal{E} to be the finite set of *events*, that is, points in time, at which a meeting starts or ends or a character is born or dies. Our aim is to find the smallest number λ_{OPT} of permutations that accommodates all events subject to the constraints above. This also minimizes the number of block crossings bc_{OPT} since $\text{bc}_{\text{OPT}} = \lambda_{\text{OPT}} - |\mathcal{E}'| + 1$, where \mathcal{E}' denotes the points in time at which at least one character is born or dies (including the birth of the first character and death of the last character in the storyline).

Our Results. Partly inspired by the ILP from Gronemann et al. [8], we developed a SAT formulation of the problem that can be used to decide whether there is a solution using a fixed number of permutations (and, hence, block crossings); see Sect. 2. Initial experiments with a similar ILP model performed poorly and led us to explore SAT solvers. We experimentally compare our new SAT approach to the two exact algorithms from our previous paper [3]; see Sect. 3. The source code of all three implementations is available online².

² <http://www1.pub.informatik.uni-wuerzburg.de/pub/data/storylines/>.

2 SAT Formulation for the Decision Problem

We present a SAT formulation that encodes, for a given storyline \mathcal{S} and an integer λ , whether there is a solution whose sequence of permutations consists of exactly λ elements. From a satisfying truth assignment we can derive the solution for \mathcal{S} . The optimal number of block crossings can then be found using this decision problem by searching for the minimum satisfiable λ , for example using linear or exponential search. Our formulation is inspired by the ILP of Gronemann et al. [8], which minimizes the number of pairwise crossings in a storyline visualization.

In the following, we do not always describe the clauses in conjunctive normal form, using other operators where this improves readability. The transformation into conjunctive normal form is straightforward. For the sake of completeness, the result of this transformation is shown in the full version [4, Appendix A]. In the following, unless specified or bound otherwise, the variables and clauses are quantified over all $i, j, k \in C$ with $i \neq j$, $i \neq k$, $j \neq k$ and all $r, p \in \{1, \dots, \lambda\}$, $r \neq p$, and $\ell \in \{1, \dots, \mu\}$, where μ is the number of *meeting groups*, a concept we introduce later on.

Describing the Permutations. To describe a solution, we start with the sequence of permutations $\Pi = [\pi_1, \dots, \pi_\lambda]$. Each permutation π_r is represented by Boolean variables of type x_{ij}^r . These variables describe the relative order of the characters in the permutation. The truth assignment of variable x_{ij}^r indicates whether character i is above character j in permutation π_r . To handle “dead” characters, we introduce another set of variables o_i^r . Character i is omitted in permutation π_r if and only if o_i^r is true. The clauses described here and under the following two headers (constraints for permutations, crossings between characters, and block crossings) are only active if all involved characters are available (that is, $\neg o_i^r$) in the permutation considered. We model this by adding o_i^r as a positive literal to each clause for each affected character i .

To ensure that the variables describe a permutation, we add the following clauses. We guarantee antisymmetry by $x_{ij}^r \Leftrightarrow \neg x_{ji}^r$. We ensure transitivity by $x_{ij}^r \vee x_{jk}^r \vee x_{ki}^r$ and $\neg x_{ij}^r \vee \neg x_{jk}^r \vee \neg x_{ki}^r$; this forces one of the three variables to have a different value than the others.

Crossings Between Characters. To simplify the treatment of crossings, we introduce variables that indicate when they occur. For $r \in \{1, \dots, \lambda - 1\}$, variable χ_{ij}^r encodes whether characters i and j have a crossing between permutations r and $r + 1$. This is the case precisely if they change their relative order between the two permutations, that is: $\chi_{ij}^r \Leftrightarrow (x_{ij}^r \neq x_{ij}^{r+1})$ for all $r \in \{1, \dots, \lambda - 1\}$. Note that this – together with the previously described clauses – implies $\chi_{ij}^r \Leftrightarrow \chi_{ji}^r$. (Recall that constraints involving omitted characters are “switched off” using the variables of type o_i^r).

According to our problem definition we have to ensure that if there is an addition or removal of characters between successive permutations, then there can be no block crossing. So we forbid crossings for all pairs of characters between

permutation π_r and π_{r+1} if a character i is added or removed between these permutations: $(o_i^r \neq o_i^{r+1}) \Rightarrow \neg\chi_{jk}^r$ for all $r \in \{1, \dots, \lambda - 1\}$.

Block Crossings. By the problem definition, there is at most one block crossing between any two successive permutations π_r and π_{r+1} . We describe this block crossing by partitioning the character set of permutation π_r into three sets F_r , G_r , and H_r . For simplicity, we drop the subscript r in the following. We express the membership of a character i in any of these sets using variables f_i^r , g_i^r and h_i^r , respectively. Let G and H be the two sets of characters that are involved in the potential block crossing between π_r and π_{r+1} , and let F be the set of characters that are not affected by the crossing. If there is no block crossing between the two permutations, at least one of the two sets G and H is empty.

First we add clauses that ensure that every character is in one of the three sets, that is, exactly one of the variables f_i^r , g_i^r , and h_i^r is true. Next, the characters of G and the characters of H must each form a contiguous block. We enforce this by requiring that a character j is in G if j lies between two characters i and k in G : $x_{ij}^r \wedge x_{jk}^r \wedge g_i^r \wedge g_k^r \Rightarrow g_j^r$. Similarly, for H we require $x_{ij}^r \wedge x_{jk}^r \wedge h_i^r \wedge h_k^r \Rightarrow h_j^r$.

We ensure that the blocks G and H are adjacent, by requiring that no character in F lies between characters in G and in H : $x_{ij}^r \wedge x_{jk}^r \wedge g_i^r \wedge h_k^r \Rightarrow \neg f_j^r$. Additionally, we prescribe the order of the blocks G and H in the permutation by restricting the characters in G to be above the characters of H : $g_i^r \wedge h_j^r \Rightarrow x_{ij}^r$.

Finally, we ensure that two characters cross each other if and only if they participate in the block crossings, that is, if one of the characters is in G and the other is in H : $g_i^r \wedge h_j^r \Leftrightarrow \chi_{ij}^r$ for all $r \in \{1, \dots, \lambda - 1\}$.

Meeting Groups. So far we have introduced various structural constraints to our variables, but we haven't yet established the connection to our input storyline \mathcal{S} . We implement this connection now through the concept of *meeting groups*. A meeting group is a set of meetings that contain a common point in time. Instead of the meeting triples (that is, a set of characters, start time, and end time), we only consider the character sets for the meeting group. Characters who are alive at that time, but are not part of any meeting, are added to the meeting group as a singleton meeting.³ We transform the storyline \mathcal{S} to a sequence of meeting groups $\mathcal{M} = [\mathcal{M}_1, \dots, \mathcal{M}_\mu]$ by sorting the events in \mathcal{E} and putting together the meetings and live characters for each event in the correct order. We use \mathcal{M} only to construct our SAT instance; afterward we transform the satisfying assignment back into a solution for \mathcal{S} .

We add variables that connect these meeting groups to the permutations of the solution. The variable q_ℓ^r indicates whether the meeting group \mathcal{M}_ℓ is assigned to permutation π_r . We require that every meeting group is assigned to exactly one permutation, that is, every group is assigned somewhere ($\bigvee_{r=1}^\lambda q_\ell^r$) and no group is assigned twice ($\neg(q_\ell^r \wedge q_\ell^p)$).

The meeting groups must be assigned to permutations in the correct order. If we map \mathcal{M}_ℓ to π_r , $\mathcal{M}_{\ell-1}$ has to be assigned to the same permutation or an

³ This concept of meeting groups is similar to the trees constructed by Gronemann et al. [8] to generate MLCM-TC instances.

earlier one: $q_\ell^r \Rightarrow \bigvee_{j=1}^r q_{\ell-1}^j$ for $\ell \in \{2, \dots, \mu\}$. We can assume that the first meeting group is assigned to the first permutation, as it is not optimal to use block crossings before the first meetings. Therefore, we set q_1^1 to true.

Next, we handle the birth and death of characters. Let \mathcal{L}_i be the meeting groups that contain character i . A permutation π_r should contain exactly the characters that are contained in the assigned meeting groups: those are precisely the alive characters. We add the clause $q_\ell^r \Rightarrow \neg o_i^r$ if $\mathcal{M}_\ell \in \mathcal{L}_i$ and the clause $q_\ell^r \Rightarrow o_i^r$ if $\mathcal{M}_\ell \notin \mathcal{L}_i$. This makes sure that characters involved in meetings must be present and dead characters are omitted.

Note that we allow permutations to not have any meeting groups assigned to them. This is necessary, for example to allow multiple block crossings between successive meetings (which may be necessary in an optimal drawing [3]). However, such “loose” permutations can be exploited to avoid block crossings by omitting all characters for one permutation and reintroducing them afterward in an arbitrary order. To forbid this, for $r = 2, \dots, \lambda$, if no meeting group is assigned to permutation π_r , we do not allow characters to be removed or added in π_r : $\bigwedge_{\ell=1}^\mu \neg q_\ell^r \Rightarrow (o_i^r = o_i^{r-1})$.

Finally, we come to the actual storyline visualization constraint: characters in a meeting must form a contiguous group in the corresponding permutation. We add clauses that prohibit characters that are not part of a meeting from being between characters in the meeting. That is, if characters i and k are part of a certain meeting in \mathcal{M}_ℓ and j is not, we have $q_\ell^r \Rightarrow (x_{i,j}^r = x_{k,j}^r)$.

This concludes our SAT formulation. If the resulting formula has a satisfying assignment, a solution to our storyline block crossing minimization problem exists, and it is easy to extract the permutations. To get the function A that maps the time to the permutations, we have to remember which meeting group corresponds to which point in time.

Counting the quantifiers in the above construction shows that there are $O(\lambda(\kappa^2 + \mu))$ variables and $O(\lambda\mu(\lambda + \kappa^3))$ clauses. The conjunctive normal form of this SAT formula can clearly be constructed from the storyline in polynomial time.

3 Experimental Evaluation

We refer to the approach from Sect. 2 as SAT. Additionally, we have implemented two exponential-time exact algorithms that minimize block crossings [3]. The first is a branching algorithm that searches for the shortest sequence of block crossings using iterative deepening depth-first search (ITD). This search strategy ensures low memory usage. The second algorithm is fixed-parameter tractable in the number of characters and works by performing a breadth-first search in an exponentially-large state graph. Note that these two algorithms do not support concurrent meetings, whereas SAT does. We also consider an algorithm by Gronemann et al. [8] that optimizes pairwise crossings.

Implementation Details. All implementations are written in C++, with the exception of some “driver” code in Python for SAT. Comparable effort has been put into optimizing each program. Memory usage was not optimized, but there are no flagrant memory inefficiencies.

SAT uses the SAT formulation from Sect. 2 and performs exponential search on λ . We use Python to write CNF SAT instances in DIMACS format, to run `minisat` [5, 6] on these instances, and to perform the search; the exponential search uses factor 2. We have used version 2.2.0 of `minisat`.⁴ As runtime of SAT, we report the total time spent by `minisat`. This includes all “real” work, as well as launching `minisat` for each formula and the time it spends reading the DIMACS files; it does not include the runtime of our Python code, which has unnecessarily-poor performance and would be unfair in comparison to the other algorithms.

ITD and FPT are implemented in C++ following the description in [3], including the data structure for block crossings and checking meetings. For ITD, we branch and “unbranch” on a single data structure rather than making copies. FPT performs a breadth-first search in a large graph. We store the nodes explicitly in a flat array addressed by Lehmer codes [10]: this requires $\Theta(\kappa!n)$ space, but enables efficient lookup. The edges of the graph are enumerated lazily using the “forward pointers” from the original paper.

All runtime experiments have been performed on an Intel[®] Core™ i5-2400 CPU at 3.10 GHz with 8 GB of RAM and running Windows 7. This configuration is in some contrast to the experimental setup of Gronemann et al.: a 2×10 -core machine with 128 GB of RAM. Our implementations are single-threaded; their implementation, being based on CPLEX, presumably makes use of the available cores, but this is not reported explicitly.

Real-World Instances. We use the same real-world instances as Gronemann et al. [8]. These include three movies and chapters from several books. See Fig. 1

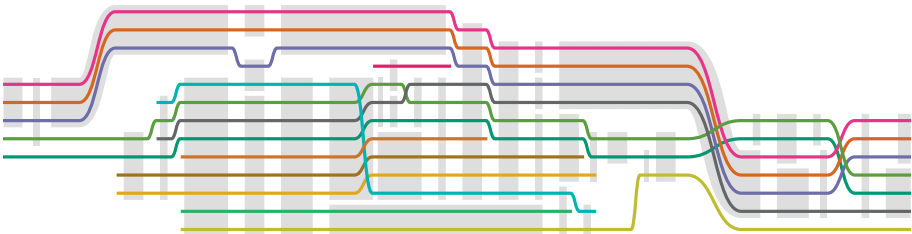


Fig. 1. A snippet of a block-crossing optimal drawing of *The Matrix* based on the sequence of permutations found by SAT, and the start and end times of the meetings (visualized by the gray blocks). The drawing reflects the linear order of the events but not their absolute points in time.

⁴ Slightly modified to measure peak memory usage on Windows.

for a block-crossing optimal drawing of The Matrix computed using SAT. More drawings computed using SAT are found in the full version [4, Appendix B].

Table 1 compares our block crossing optimization to solutions optimized for pairwise crossings. It shows that the optimal number of block crossings is much lower than the optimal number of pairwise crossings. This decrease is not just counting things differently: Gronemann et al.’s drawing of The Matrix, for example, has the optimal number of 12 crossings and happens to have 8 block crossings, whereas we give an optimal drawing with 4 block crossings. Our drawing happens to have 33 pairwise crossings: this presents an interesting trade-off.

Table 1. Comparison of pairwise crossings (cr) and block crossings (bc) on movie instances; subscript OPT indicates the value that the algorithm optimized. The runtime of both approaches is similar, even on rather different machines (see Sect. 3 – “Implementation Details”).

Instance	Block crossings using SAT					Gronemann et al. [8]		
	λ_{OPT}	Memory	cr	bc _{OPT}	Time [s]	cr _{OPT}	bc	Time [s]
Star Wars	20	79 MB	54	10	3.77	39	18	0.99
The Matrix	18	67 MB	21	4	2.86	12	8	0.77
Inception	23	39 MB	51	12	1.54	35	20	2.02

The book instances unfortunately present a strong challenge for our algorithms. Even though there are no concurrent meetings, the number of characters immediately disqualifies FPT and the optimum is too large for ITD. This leaves SAT, but these instances (as modeled in Gronemann et al.) contain an extreme number of ‘births’ and ‘deaths.’ While this is convenient for their algorithm (or at least: not detrimental), our SAT formulation requires a large number of permutations to handle this. One might hope that – even though large – these formulas are still relatively easy for `minisat`: alas, they are not.

Finally, we look at the exponential search that SAT uses to find the optimal number of permutations. If testing a number of permutations takes exponential time (we are solving a SAT instance, after all), a single overestimate would be disastrous. However, on the real-world instances we observe fairly modest time for overestimated λ (see Fig. 2). This means exponential search can have a significant advantage over linear search. On the movie instances, using exponential search is indeed faster than linear search, but just by about a third.

Random Instances. We test using random instances of two kinds. The first are *uniform* instances and these are the same as in previous work [3]. First, pick κ , n , and a probability p . (We report here on $p = 0.5$.) Then generate a meeting by picking, independently at random with probability p , whether each character is in the meeting. Reject meetings with fewer than two characters, and repeat until there are n meetings. We let all characters be alive at all times, so we can run all three algorithms.

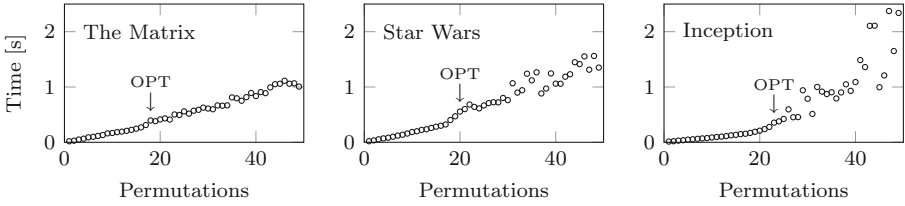


Fig. 2. Runtime of `minisat` for different numbers of permutations on the movie instances. Recall that the number of permutations does not equal the number of block crossings.

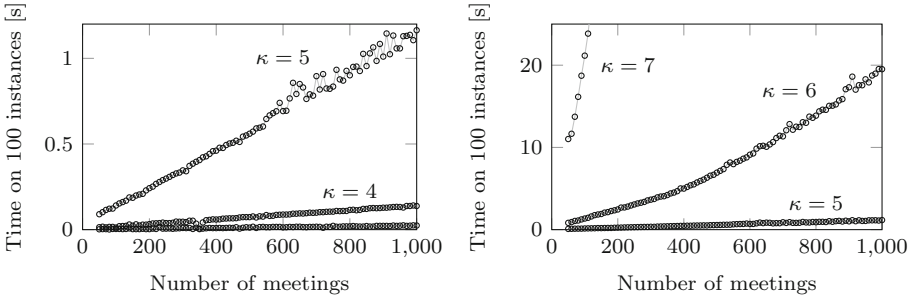


Fig. 3. Total runtime of FPT on 100 random instances from the uniform model with $p = 0.5$, increasing number of meetings, and $\kappa \in \{3, 4, 5, 6, 7\}$.

Figure 3 shows the runtime of FPT on these instances as a function of n , for various numbers of characters. It confirms the fixed-parameter tractable runtime in practice. Note that the plot reports the runtime for solving 100 instances. The other algorithms have trouble handling 1000 meetings in any reasonable setting; with $\kappa = 5$, FPT solves 100 such instances in little more than a second. However, the explosive dependence on κ is also clear.

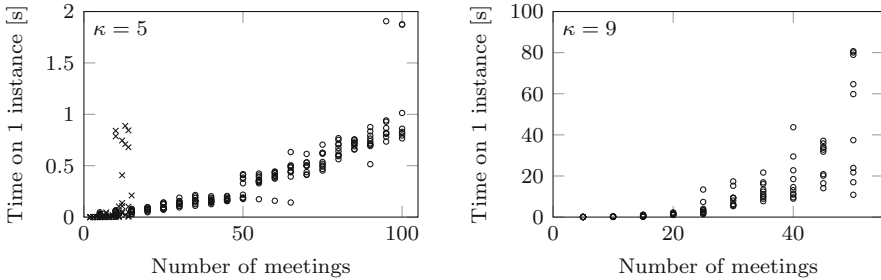


Fig. 4. Circular marks: runtime of SAT on instances from the uniform model with $p = 0.5$, and $\kappa = 5$ (left) and $\kappa = 9$ (right). Crosses, left plot: runtime of ITD. It is highly variable and practically dominated by SAT.

Figure 4 similarly shows the runtime of SAT. Since there is more variance, we show 10 data points for every number of meetings, rather than the sum. For $\kappa = 5$, SAT can easily handle 100 meetings: except for an outlier, we are not yet hit by a runtime explosion. Note, however, that it is significantly slower than FPT: approximately three orders of magnitude at $n = 100$. As for ITD: it is so slow on these instances that its runtime escapes the plot almost immediately.

For $\kappa = 9$, a different picture develops. Firstly, SAT experiences difficulty as the number of meetings increases. With instances approaching 50 meetings, the runtime starts to explode. For these instances, the runtime of FPT is similar since it too has become slow at $\kappa = 9$. The difference is that, if we are willing to wait longer, SAT can be run on instances with more than 9 characters, whereas FPT is quite fundamentally limited by its memory usage (see “Memory Usage”).

Yet another picture emerges when we look at instances that have a solution with few block crossings. To consider such instances is fair since in practice we are particularly interested in instances that can be realized with few block crossings. First, pick κ , n , a probability p , and a number β : we generate random instances that have optimum at most β as follows. Start from the identity permutation and sample β uniformly-random block crossings: this results in a sequence of $\beta + 1$ permutations. Now generate n meetings: pick, for each one independently, one of the permutations at random and then c adjacent characters from this permutation at random, where c is binomially distributed with success probability p so as to match the uniform model; put these meetings in the order of the permutations they come from. By construction, these instances have a solution with (at most) β block crossings.

Figure 5 shows that SAT and ITD can solve much larger instances of this kind. This is as expected, since β directly bounds the branching depth of ITD and the number of permutations required by SAT. We see that SAT practically dominates ITD; the only reason to use ITD is if no high-quality SAT solver is available, or if memory usage is important (the redeeming quality of ITD).

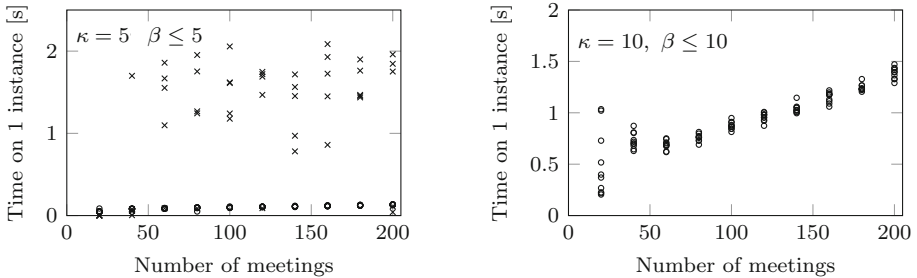


Fig. 5. Circular marks: runtime of SAT on random instances from the small-opt model with $p = 0.5$, and $\kappa = 5$ (left) and $\kappa = 10$ (right). Crosses, left plot: runtime of ITD. It is highly variable and practically dominated by SAT.

Memory Usage. Table 2 shows that the (peak) memory usage of the algorithms is quite different. ITD is implemented to branch with a single data structure. This is good for runtime (no copying), and has the additional benefit that memory usage is very low. In fact, it is hardly impacted by recursion depth since we use iterative deepening depth-first search and, rather than having the entire data structure at each level of the recursion, there is only a small stack frame.

The memory usage of SAT increases significantly with λ and the overhead is more than for ITD: this is because of the large, explicitly-stated SAT formulas and the use of a general-purpose SAT solver.

For small κ , FPT uses less memory than SAT due to the latter's overhead. However, FPT clearly uses the most memory as κ increases, since it quite fundamentally relies on the memoization of a large recurrence. Its memory usage in fact limits the number of characters that can be supported – in practice by the available memory, but even more generally by the memory architecture of normal environments.

Table 2. Memory usage in MB on uniform random instances ($p = 0.5$) with 100 meetings and a variable number of characters κ . Only SAT's memory usage varies considerably over different instances with the same number of characters.

κ	ITD	SAT	FPT
5	0.69	13–31	1.2
6	0.69	44–48	1.7
7	0.69	64–76	8.3
8	0.69	110–218	64.0
9	0.70	338–422	645.9
1000	4.00	×	×

Concluding Remarks. We conclude with some practical advice about picking an algorithm. The first consideration is a hard constraint: if concurrent meetings are required, ITD and FPT are disqualified and SAT remains as a fine default. We now assume concurrent meetings are not required.

If the number of characters is small, the use of FPT is clearly preferred. This algorithm can truly be considered fixed-parameter tractable in the number of characters κ . However, the dependence on κ includes factorial space, which makes it impractical to run the algorithm on personal computers beyond $\kappa = 10$ and impossible to run at all for even a few characters more than that. Many real-world instances have too many characters for FPT.

The runtime of both ITD and SAT depends heavily on the number of block crossings in the optimum. For very small optimum, ITD can be faster, but only if the number of characters is also quite small: there is still the branching factor of $\kappa!$. If memory is a problem and the optimum is small, then ITD is an option, but in general SAT is vastly preferable.

As a final remark, we note that all these implementations are single-threaded, and as such achieve only “25%” utilization on our quad-core test machine. ITD could be trivially parallelized by dividing the search space; FPT is trickier to parallelize from an engineering perspective. It would be possible to use a parallelized SAT solver, like PMSat [7] or HordeSAT [1]. However, it is not clear a priori how effective those would be for our specific SAT formulas.

4 Conclusion

In this paper we have presented a SAT-based algorithm for computing block-crossing optimal storyline visualizations and extensive experimentation on random instances. We have demonstrated that on some real-world instances (in particular, the movies), SAT has runtime similar to the ILP of Gronemann et al., who optimize pairwise crossings. For other instances (the books), SAT fares poorly. We have also evaluated implementations of two further algorithms for storyline block crossing optimization.

For future work, it would be interesting to perform further algorithm engineering on SAT. In particular, it may be possible to handle the birth/death of characters more efficiently or to better integrate with SAT algorithms.

In a different direction, one might use an ILP solver on a model very similar to that of Sect. 2. This would, for example, enable us to minimize the number of pairwise crossings subject to the number of block crossings being optimal. However, preliminary experiments showed very poor performance.

From a graphic design perspective, optimizing for block crossings intuitively makes sense. However, we are not aware of any user studies that investigated whether block crossings are good, and what the trade-offs are. For example, is a 4×4 block crossing equally bad as a 2×8 block crossing?

Acknowledgments. We thank Martin Gronemann for providing the input files used in the experiments of [8].

References

1. Balyo, T., Sanders, P., Sinz, C.: HordeSat: a massively parallel portfolio SAT solver. In: Heule, M., Weaver, S. (eds.) SAT 2015. LNCS, vol. 9340, pp. 156–172. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24318-4_12
2. Bekos, M.A., Kaufmann, M., Zielke, C.: The book embedding problem from a SAT-solving perspective. In: Di Giacomo, E., Lubiw, A. (eds.) GD 2015. LNCS, vol. 9411, pp. 125–138. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27261-0_11
3. van Dijk, T.C., Fink, M., Fischer, N., Lipp, F., Markfelder, P., Ravsky, A., Suri, S., Wolff, A.: Block crossings in storyline visualizations. *J. Graph Algorithms Appl.* **21**(5), 873–913 (2017). <https://doi.org/10.7155/jgaa.00443>
4. van Dijk, T.C., Lipp, F., Markfelder, P., Wolff, A.: Computing storyline visualizations with few block crossings. arXiv report <http://arxiv.org/abs/1709.01055> (2017)

5. Eén, N., Mishchenko, A., Sörensson, N.: Applying logic synthesis for speeding up SAT. In: Marques-Silva, J., Sakallah, K.A. (eds.) SAT 2007. LNCS, vol. 4501, pp. 272–286. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72788-0_26
6. Eén, N., Sörensson, N.: MiniSat SAT solver (2003). <http://minisat.se>
7. Gil, L., Flores, P., Silveira, L.M.: PMSat: a parallel version of MiniSAT. *J. Satisf. Bool. Model. Comput.* **6**, 71–98 (2008). <https://satassociation.org/jsat/index.php/jsat/article/view/70>
8. Gronemann, M., Jünger, M., Liers, F., Mambelli, F.: Crossing minimization in storyline visualization. In: Hu, Y., Nöllenburg, M. (eds.) GD 2016. LNCS, vol. 9801, pp. 367–381. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_29
9. Kim, N.W., Card, S.K., Heer, J.: Tracing genealogical data with TimeNets. In: Proceedings of the International Conference on Advanced Visual Interfaces (AVI 2010), pp. 241–248 (2010). <https://doi.org/10.1145/1842993.1843035>
10. Knuth, D.E.: *The Art of Computer Programming, Volume 3: Sorting and Searching*, 2nd edn. Addison Wesley Longman Publishing Co., Inc., Redwood City (1998)
11. Kostitsyna, I., Nöllenburg, M., Polishchuk, V., Schulz, A., Strash, D.: On minimizing crossings in storyline visualizations. In: Di Giacomo, E., Lubiw, A. (eds.) GD 2015. LNCS, vol. 9411, pp. 192–198. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27261-0_16
12. Munroe, R.: Movie narrative charts (2009). <https://xkcd.com/657/>. Accessed 16 Feb 2017
13. Tanahashi, Y., Ma, K.: Design considerations for optimizing storyline visualizations. *IEEE Trans. Vis. Comput. Graph.* **18**(12), 2679–2688 (2012). <https://doi.org/10.1109/TVCG.2012.212>
14. Wertheimer, M.: Untersuchungen zur Lehre von der Gestalt. II. *Psychologische Forschung* **4**(1), 301–350 (1923)

MLSEB: Edge Bundling Using Moving Least Squares Approximation

Jieting Wu, Jianping Zeng, Feiyu Zhu, and Hongfeng Yu^(✉)

University of Nebraska-Lincoln, Lincoln, NE 68588, USA
{jwu, jizeng, fzhu, hfyu}@cse.unl.edu

Abstract. Edge bundling methods can effectively alleviate visual clutter and reveal high-level graph structures in large graph visualization. Researchers have devoted significant efforts to improve edge bundling according to different metrics. As the edge bundling family evolve rapidly, the *quality* of edge bundles receives increasing attention in the literature accordingly. In this paper, we present MLSEB, a novel method to generate edge bundles based on moving least squares (MLS) approximation. In comparison with previous edge bundling methods, we argue that our MLSEB approach can generate better results based on a quantitative metric of quality, and also ensure scalability and the efficiency for visualizing large graphs.

Keywords: Edge bundling · Graph visualization
Moving least squares · Visualization quality

1 Introduction

Traditional exploration methods of large graphs are often overwhelmed by severe visual clutter such as excessive vertex overlappings and edge crossings. *Edge bundling* is one of the effective approaches to reducing edge crossings in graph drawings. The main idea of edge bundling is to visually merge edges with similar features (e.g., position, direction, and length) such that edge crossings are significantly reduced and the readability of graph drawings is improved.

Substantial efforts have been made to develop various edge bundling algorithms to improve visual results. The current edge bundling family have provided a diverse graph layouts that work with a wide spectrum of applications and domains based on different strategies or metrics [22]. As the edge bundling techniques develop rapidly, the information visualization community is putting increasing interests in evaluating the results of edge bundle drawings. The readability and faithfulness criteria are often used to evaluate graph drawings. Edge bundling helps simplify graph drawings and increase readability, but yields distortion that makes it hard to preserve the faithfulness of original graphs [29]. To holistically address the evaluation of both readability and faithfulness for edge bundling visualization, Lhuillier et al. [22] suggested a general metric where a ratio of clutter reduction to amount of distortion is computed to measure the

quality of edge bundling visualization. In this work, we aim to generate high-quality edge bundling results based on Lhuillier’s suggestion, and meanwhile ensure scalability and efficiency.

We introduce a novel edge bundling technique to generate edge bundles with moving least squares (MLS) approximation, namely MLSEB. Inspired by thinning an unorganized point cloud to curve-like shapes [20], we use a distance-minimizing approximation function to generate bundle effects. In particular, we first sample a graph into a point cloud data, and then use a moving least squares projection to generate curve-like bundles. Based on Lhuillier’s suggestion, we develop a quality assessment to evaluate edge bundling results. Using different real-world datasets, we demonstrate that MLSEB can produce bundle results with a higher quality, and is scalable and efficient for large graphs by comparing different edge bundling methods.

2 Related Work

The recent study [22] has surveyed the state-of-the-art edge bundling techniques and their applications in a very detailed manner. We revisit some of these methods by briefly summarizing the categories of the diverse bundling techniques. We consider our method as an image-based method, and hence we will discuss the image-based methods in more details. We will also cover some studies of quality evaluation in edge bundling and some studies on moving least squares approximation.

Holten [11] pioneered the edge bundling techniques in graph drawings using a hierarchical structure. *Geometric-based* methods [4, 17, 18, 25] used a control mesh to guide bundling process. *Energy-based* minimization methods have been also used in many studies. Examples include ink-minimization methods [8, 9] and force-directed methods [12, 31, 37, 42, 43]. Most of these methods used compatibility criteria to measure the similarity of different edges based on spatial information (i.e., length, position, angle, and visibility), and then moved the similar edges with ink-minimization or force-directed strategies.

Image-based techniques used a density assessment to guide bundling process [3, 6, 13, 23, 33, 44]. These methods are generally based on *Kernel Density Estimation*. Kernel density estimation edge bundling (KDEEB) [13] first transformed an input graph into a density map using kernel density estimation, and then moved the sample points of edges towards the local density maxima to form bundles. Peysakhovich et al. [33] extended KDEEB using edge attributes to distinguish bundles. CUDA Universal Bundling (CUBu) [44] used GPU acceleration to enable interactively bundling a graph with a million edges. Fast Fourier Transform Edge Bundling (FFTEB) [23] improved the scalability of density estimation by transforming the density space to the frequency space.

There are other edge bundling studies. Bach et al. [2] investigated the connectivity of edge bundling methods on Confluent Drawings. Nguyen et al. [30] proposed an edge bundling method for streaming graphs, which extended the idea of TGI-EB [31]. Wu et al. [41] used textures to accelerate bundling for

web-based applications. Kwon et al. [16] showed their layout, rendering, and interaction methods for edge bundling in an immersive environment.

Several studies introduced general metrics to quantify the readability [5, 35, 36, 38] and the faithfulness [29] of graph drawings. Some existing studies in edge bundling have defined quality assessments to evaluate the resulting bundles. Nguyen et al. [29] conducted a study on the faithfulness for force-directed edge bundling methods. Telea et al. [39] surveyed different hierarchical edge bundling techniques and conducted a user study for the comparison between bundled and unbundled methods. Pupyrev et al. [34] and Kobourov et al. [15] worked towards measuring edge crossings. KDEEB [13] and CUBu [44] proposed post-relaxation if the distortion of edge bundles is too large, such that the mental map is preserved. For sequence graph edge bundling, Hurter et al. [14] used interpolation to preserve the mental map between sequence graphs. McGee and Dingliana [26] conducted an empirical study on the impact of edge bundling.

Moving least squares (MLS) has been widely used to approximate smooth curves and surface from unorganized point clouds [1, 21, 27]. Lee [20] constructed a curve-like shape from unorganized point clouds using an Euclidean minimum spanning tree. Least square projection (LSP) has been used in graph drawings [32], where multidimensional data points are projected into lower dimensions, while the similar relationship in neighboring points is preserved.

3 Background

3.1 Definition of Edge Bundling

We first revisit a formal definition of edge bundling [23]. Let $G = (V, E) \subset \mathbb{R}^2$, $V = \{v_i\}$, $E = \{e_i\}$ be a graph, where v_i is a vertex and e_i is an edge of G . Let $D : E \rightarrow \mathbb{R}^2$ be a drawing operator, such that $D(G)$ represents the drawing of G and $D(e_i)$ represents the drawing of an edge e_i . We define a compatibility operator ϕ , where $\phi(e_i, e_j)$ measures the similarity of two edges e_i and e_j . Edges that are more similar than a threshold ϕ_{max} should be bundled together, and ϕ can be used with some reasonable attributes and metrics (e.g., spatial information [12]). Let $B : \mathcal{D} \rightarrow \mathcal{D}$ be a bundling operation, where $\mathcal{D} \subset \mathbb{R}^2$ denotes the space of all graph drawings, and $B(D(e_i))$ denotes the resulting bundled drawing of e_i . For example, $D(e_i)$ can be a straight line drawing and $B(D(e_i))$ can be a drawing of curve or polyline. Hence, an edge bundling algorithm can be expressed as:

$$\begin{aligned} \forall (e_i \in G, e_j \in G) | \phi(e_i, e_j) < \phi_{max} \rightarrow \\ \delta(B(D(e_i)), B(D(e_j))) \ll \delta(D(e_i), D(e_j)), \end{aligned} \quad (1)$$

where δ is a distance metric in \mathbb{R}^2 . Different edge bundling approaches explored various ϕ , B , and δ to tackle Eq. 1 to gain different visual effects of edge bundling [22].

3.2 Quality of Edge Bundling

Edge bundling techniques trade the increase of readability for overdrawing by bending edges to form bundle effects. Hence, edge bundle techniques naturally generate distortion from original graphs. To quantify the quality of a bundled graph, Lhuillier et al. [22] suggested to use the ratio of *clutter reduction* C to *amount of distortion* T as a quality metric Q , i.e.,

$$Q = \frac{C}{T}, \quad (2)$$

In general, a larger Q corresponds to a higher quality, and vice versa. Lhuillier et al. [22] further posed a distortion measure. Simply, for an edge e_i , the distortion between an unbundled drawing $D(e_i)$ and a bundled result $B(D(e_i))$ is measured by computing the distance between them, i.e., $\delta(D(e_i), B(D(e_i)))$. Therefore, the overall distortion T between an original unbundled graph and its bundled result can be defined as:

$$T = \sum_{i=1}^n \delta(D(e_i), B(D(e_i))), \quad (3)$$

where n is the number of edges. Equation 3 provides an intuitive metric to evaluate the distortion generated by a bundled graph. The calculation of clutter reduction has not been fully concluded in the existing work. We propose a simple method to evaluate clutter reduction C , modify Eq. 3 to compute T , and then use C and T to quantify the quality Q of edge bundling (Sect. 6.2).

4 Our Bundling Algorithm

The main purpose of edge bundling is to achieve appealing bundle effects by bending edges, expressed by Eq. 1. Meanwhile, according to Eq. 2, an ideal algorithm should increase clutter reduction C , while decrease amount of distortion T , in order to achieve a higher quality Q of edge bundling. Therefore, we should holistically address Eqs. 1 and 2, which, however, has not been fully investigated in the existing work [22].

4.1 Sampling

In general, given a graph G , a polyline is used to draw the line or curve presentation of an edge e_i . Sample points x_k^i , namely *sites*, are used to discretize the drawing of e_i . Formally,

$$\{x_k^i | 1 \leq k \leq m_i\} \approx D(e_i), \quad (4)$$

where m_i is the number of sites for $D(e_i)$. Note, many methods [13, 23, 33, 44] use a sampling step that is a small fraction of the size of the display to sample

each edge, which means the number of sites of $D(e_i)$ may be different. Similarly, the bundled drawing can also be discretized as:

$$B(\{x_k^i | 1 \leq k \leq m_i\}) \approx B(D(e_i)). \quad (5)$$

We measure the distortion between $D(e_i)$ and $B(D(e_i))$ by summing the Euclidean distance between each pair of x_k^i and $B(x_k^i)$. Let $|\cdot|$ denote the Euclidean distance. Replace the edges in Eq. 3 using Eqs. 4 and 5, we have

$$T = \sum_{i=1}^n \left(\sum_{k=1}^{m_i} |x_k^i, B(\{x_k^i\})| \right). \quad (6)$$

Similarly, Eq. 1 can be modified as:

$$\begin{aligned} \forall (e_i \in G, e_j \in G) | \phi(e_i, e_j) < \phi_{max} \rightarrow \\ |B(\{x_k^i\}), B(\{x_k^j\})| \ll |\{x_k^i\}, \{x_k^j\}|. \end{aligned} \quad (7)$$

Therefore, we discretize each edge drawing $D(e_i)$ of G by Eq. 4. All the sample points generated by Eq. 4 form a point cloud. According to Eq. 7, x_k^i is moved to a new position $B(x_k^i)$ by a bundling operator B . In the case of kernel density estimation edge bundling [13, 23, 33, 44], x_k^i is moved to $B(x_k^i)$ according to its local density gradient. These methods form the bundles by gathering sample points to their local density maxima, but do not consider the distortion of edges when moving sample points. Therefore, certain artifacts, such as lattice effects and subsampled edge fragments, can be incurred. The methods, such as resampling and post-relaxation [13, 44], have been proposed to address these issues. However, these methods typically introduce a significant performance overhead that is challenging to alleviate [44]. We develop a new bundling operator B with respect to Eq. 7, and minimize the distortion of each sample point locally. Moreover, our method does not require resampling, and thereby can reduce the computational cost.

4.2 Moving Least Squares Approximation

We consider all the points formed by sampling, and assess the global distortion by expressing Eq. 6 as:

$$\mathcal{T} = \sum_{i=1}^S |x_i - B(x_i)|^2, \quad (8)$$

where x_i is a site in the point cloud, and S is the number of sites of all edges.

We assume there is a *skeleton* near x_i and its neighborhood locally. A skeleton can be a suitable place to gather curves to form bundles [6]. Assume a skeleton can be interpreted as an implicit polynomial or piece-wise polynomial curve f_i , which is unknown. The unknown f_i can be gained by computing the coefficients

of f_i , i.e., by minimizing the following weighted least squares error ϵ within a set $\mathcal{H}(x_i)$ consisting of x_i and its neighbor sites:

$$\epsilon = \sum_{j=1}^{h_i} |x_j - f_i|^2 \theta(|x_j - x_i|), \tag{9}$$

where $x_i \in \mathcal{H}(x_i)$, $x_j \in \mathcal{H}(x_i)$, h_i is the size of $\mathcal{H}(x_i)$, and $|x_j - f_i|$ means the shortest Euclidean distance between x_j and f_i . We define the bundling operator B on x_i as a two-step procedure: first to construct f_i , and then to project x_i onto f_i . The projected point is thereby $B(x_i)$ that is on f_i . The distance $|x_i - B(x_i)|$ from x_i to $B(x_i)$ is locally minimized by an appropriate nonnegative weighting function θ . The input of θ is $|x_j - x_i|$, which is the distance of neighborhood x_j to the site x_i . Instead of taking all sites of a graph into account, we use a circle of radius r (*bandwidth*) centered at x_i to collect the neighborhood x_j for x_i .

If $\theta \equiv 1$, a least squares (LS) approximation is generated. However, LS approximation does not work well to generate a polynomial curve that locally reflects the density distribution of neighborhood. Alternatively, the moving least squares (MLS) method can reduce a point cloud to a thin curve-like shape that is a near-best approximation of the point set [20, 21]. Hence, we use a local assessment to approximate f_i [19]. The weighting function we use is a cubic function [27]:

$$\theta(d) = \begin{cases} 2\frac{d^3}{r^3} - 3\frac{d^2}{r^2} + 1 & \text{if } d < r, \\ 0 & \text{if } d \geq r, \end{cases} \tag{10}$$

where $d = |x_j - x_i|$. In this sense, minimizing Eq. 9 leads to an MLS approximation so that f_i is a local regression curve, and $|x_i - B(x_i)|$ is locally minimized. In other words, the distortion is locally minimized.

In our work, we use an MLS approximation to evaluate the distance $|x_j - f_i|$ for the neighborhood $\mathcal{H}(x_i)$ of x_i . Therefore, we use a basic projection [19] to construct the implicit local regression curve f_i : We take a partial derivative of Eq. 9 with respect to each coefficient of f_i , make each partial derivative equal to zero, and then solve the system of equations to generate all the coefficients of f_i [28].

Similar to existing work [6, 13, 23, 33, 44], we implement our bundling operator B through an iteration strategy. In our method, two steps are applied iteratively, as shown in Fig. 1. We initially treat x_i as $x_{i,0}$. Then, in each iteration u , the first step is to construct an optimal regression curve $f_{i,u}$ by thinning the unordered point cloud within $\mathcal{H}(x_{i,u})$, the neighborhood of $x_{i,u}$. In the second step, we project $x_{i,u}$ onto $f_{i,u}$ and obtain the projected point $x_{i,(u+1)}$, i.e., $B(x_{i,u})$. In this way, a site $x_{i,u}$ is moved to $x_{i,(u+1)}$ based on the weighting function θ of

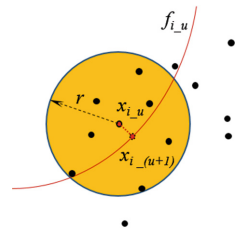


Fig. 1. Two steps of our bundling operator B on a site $x_{i,u}$ in an iteration u . First, a local implicit regression curve $f_{i,u}$ is constructed by the neighborhood of $x_{i,u}$ with a bandwidth r using the MLS approximation. Second, $x_{i,u}$ is moved to a new position $x_{i,(u+1)}$ that is the projection of $x_{i,u}$ on $f_{i,u}$.

its neighborhood $\mathcal{H}(x_{i,u})$. Different from the kernel density estimation methods [6, 13, 23, 33, 44], MLS moves the site $x_{i,u}$ in the sense that the local error ϵ is bounded with the error of a local best polynomial approximation [20]. In our current work, this process stops when the iteration number reaches a predefined threshold. Then, for each edge, we compute a *B-spline* curve based on the final positions of its sites. Figure 2 shows an example with two different iterations. For an illustration purpose, we show the corresponding *B-spline* curves for the iterations. In Fig. 2, we can see that a curve-like skeleton is gradually formed from the point cloud through the iterations in the top row, and a bundle effect becomes increasingly distinct as shown by the *B-spline* results in the bottom row.

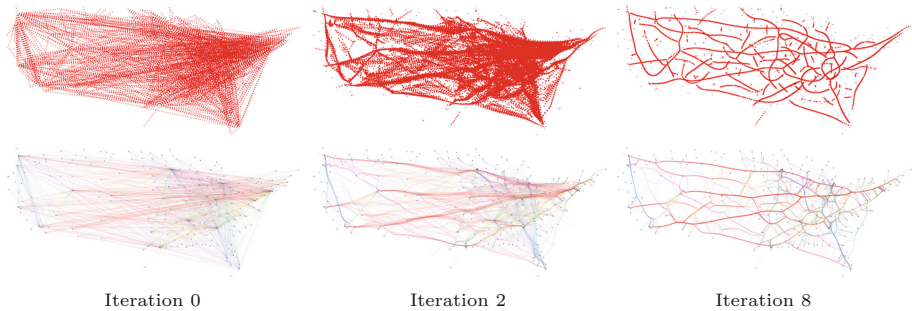


Fig. 2. Using an US airlines dataset as an example, we first sample each edge into a set of points (or sites). The resulting sites form a point cloud (top-left). The top row shows the point cloud is converged through an iterative MLS processing. The bottom row shows the corresponding B-spline results. The first column shows the initial result before MLS. The following columns show the results generated after the 2nd and 8th iteration, respectively.

Most of the existing image-based techniques use kernel density estimation (KDE), essentially, a mean-shift method that evaluates the local density maxima and advects a site based on the gradients of the local density. However, KDE does not consider the distortion (Eq. 3) when moving sample points, and thus resampling or post-relaxation is often required [13, 44]. Alternatively, our MLSEB method uses an MLS approximation that projects a site x_i to its local regression curve f_i , where f_i is locally approximated by minimizing the distance between $\mathcal{H}(x_i)$ and f_i with a weighted function (Eq. 9). Therefore, the distance between its original position x_i and its projected position $B(x_i)$ is locally minimized based on the density of its neighborhood $\mathcal{H}(x_i)$. One advantage of our method is that MLS does not need to resample each edge in bundling iterations because sites are projected into curves that do not generate over-converge artifacts or lattice effects. Fröhlich et al. [7] showed that MLS produced better convergence results than KDE in biological studies. However, it remains an open question to determine if KDE or MLS is better than one another in edge bundling. In Sect. 6.2, we will develop a quality assessment from Eq. 2, and use it to evaluate

and compare the quality of the drawings generated by our MLSEB method, the FFTEB method (a KDE-based method), and the FDEB method (a force-directed method).

5 Implementation

Our implementation involves simple data structures and computations, and thus is easy to implement. First, we sample the edges of an input graph. We use the same scheme as KDEEB's [13] to sample the input edges with a uniform step ρ . The most time consuming step in our method is gathering the neighborhood for every site. A typical solution in a GPU implementation is to use *Uniform Grid* [10] that subdivides the space into uniformly sized cells. We use this method and set the size of the cell to be $\frac{2}{3}r$ (r is a prescribed radius or bandwidth) such that we can limit the search space of each site to only cover at most 9 grid cells [10], thus avoid a $O(S^2)$ search time for S sites.

At the start of each iteration, all the sites are put into the corresponding cells according to their current positions. This can be easily parallelized using CUDA on a GPU [10]. Then, we project each site onto its local regression line. The solution to compute the coefficients of Eq. 9 is introduced in the work [19, 28]. It only requires a constant time to solve the coefficients of a linear or quadratic system of equations. This can also be parallelized using a GPU because computing the new projection position for every site is independent.

To enhance the visualization of a bundled graph, we use the same shader scheme of CUBu [44]. We use the HSVA (i.e., hue H , saturation S , value V , and alpha A) color representation to visualize edges. Each edge site x_i is encoded with an HSVA value. We encode the direction and the length of the corresponding edge into H and S , respectively. V and A are used with a parabolic profile function $c(x) = \sqrt{1 - 2|t(x) - \frac{1}{2}|}$, and $t \in [0, 1]$ is the edge arc-length parameterization. The functions of V and A are then $V(x) = \frac{l}{l_{max}} + (1 - \frac{l}{l_{max}})c(x)$ and $A(x) = \alpha(1 - \frac{l}{l_{max}} + \frac{l}{l_{max}}c(x))$ respectively, where l is the length of the edge, l_{max} is the longest edge in the graph, and α controls the overall transparency of all edges.

Next, we analyze the complexity of our MLSEB method. Similar to the existing KDE-based methods [13, 23, 33, 44], MLSEB requires gathering neighbor sites for computation. After gathering, KDE-based methods conduct kernel splatting, gradient calculation, and site advection, which use a constant time for each site. In MLSEB, the time to solve Eq. 9 and project a site to its local approximated curve is also constant for each site. Thereby, the complexity of MLSEB is the same as the traditional KDE-based methods, which is $O(I \cdot N \cdot S)$, where I is the image resolution, N is the number of bundling iterations, and S is the number of sample points. However, MLSEB does not need additional operations, such as resampling, that are employed in the existing KDE-based methods.

We explore the parameter choices of MSLEB as follows. Similar to most the existing edge bundling methods, we use a step ρ , which is 5% of the image resolution I , to sample each edge. The bandwidth, r , plays an important role in MLS to

estimate the density information around each site. A larger bandwidth captures more sample sites to reflect a more global feature, while a smaller bandwidth reveals a more local feature. By following a similar strategy in FDEB [43] and KDEEB [13], we decrease r by a reduction factor λ after each iteration. Hurter et al. [13] stated that a kernel size follows an average density estimation when $0.5 \leq \lambda \leq 0.9$. We set r to be $5\% \leq r \leq 20\%$ of the display size I to generate a stable edge-convergence result. Through a heuristic study, we found that it is sufficient to yield good results by setting the iteration number N between 3 and 10 and making the polynomial order of f_i in Eq. 9 to be 1 or 2.

6 Results

6.1 Visualization and Performance Results

We apply our MLSEB method to several graphs and compare its effect and computational performance to the two existing methods: FDEB that is the classic force-directed method, and FFTEB that is the latest enhanced KDE-based method of image-based edge bundling algorithms (such as KDEEB and CUBu).

The left column in Fig. 3 compares the visualization results of our MLSEB method with other bundling methods using the US airlines dataset (2101 edges). Our MLSEB method provides similar results, and generates tight, smooth and locally well-separated bundles. High-level graph structures are also revealed in our results. The right column in Fig. 3 shows the comparison using the US migrations dataset (9780 edges). Figure 4 shows another example using the France airlines dataset with 17274 edges. In these results, the main migration and airline patterns are clearly revealed using MLSEB. In the migrations dataset, FDEB and FFTEB fall short in showing some subtle structures of the original graph. For example, in the original node-link diagram of Fig. 3(b), the edges (within the red box) connect the city of Portland to some cities in the northern U.S are distorted significantly from their original positions in the results of the FDEB (Fig. 3(d)) and FFTEB (Fig. 3(f)), while our MLSEB result has a distinguished bundle effect that reveals this subtle graph structure. In Fig. 5, we compare the visual result of MLSEB to FFTEB using a large US migrations dataset with 545881 edges. We encode the color of a edge with only its length in this example. MLSEB shows more long-length edge patterns than FFTEB.

Table 1 shows the performance comparison between our MLSEB method and the current fastest edge bundling method FFTEB. In our performance comparison, we used the US airlines graph, the US migrations graph, the France airlines graph, and the large US migrations graph. The timing results for MLSEB and FFTEB are based on one iteration, and we excluded the timing of memory allocation and data transferring for both methods. The devices used in our experiments are a desktop with an 8X Intel Core i7-6700K 4.0 GHz CPU with 32 GB memory and a NVIDIA GeForce GTX TITAN X GPU. Comparing with the fastest algorithm FFTEB in the state-of-the-art, we can clearly see that MLSEB is at the same order of magnitude of FFTEB in terms of computational speed, as shown in Table 1.

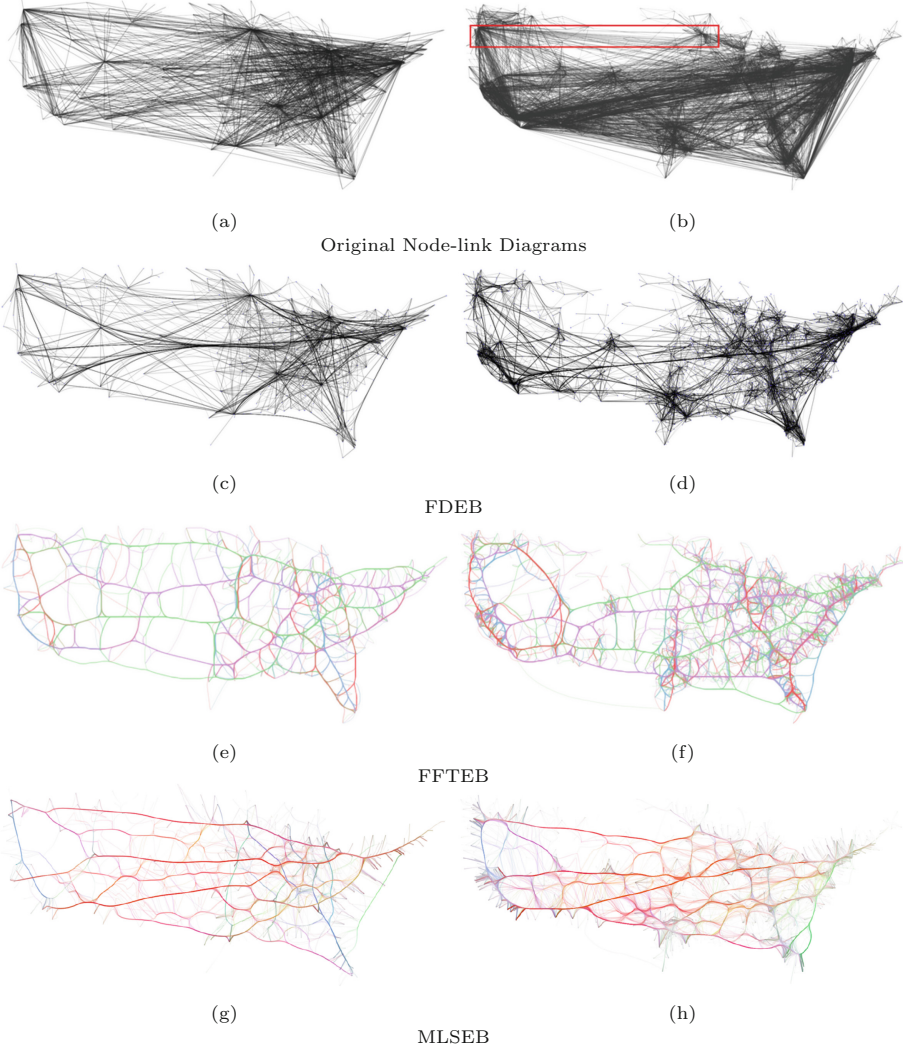


Fig. 3. Visualize the US airlines dataset (the left column) and the US migrations dataset (the right column) with three different edge bundling methods, FDEB, FFTEB and MLSEB, respectively. (Color figure online)

6.2 Quality Assessment of Bundled Graphs

Apart from comparing the visualization and performance results, we propose a quality metric to evaluate the quality of bundling drawings based on Eq. 2.

Equation 2 gives a general quality metric Q based on the ratio of clutter reduction C to amount of distortion T . However, the quantification of clutter reduction C has been not fully concluded in existing work. We propose to employ

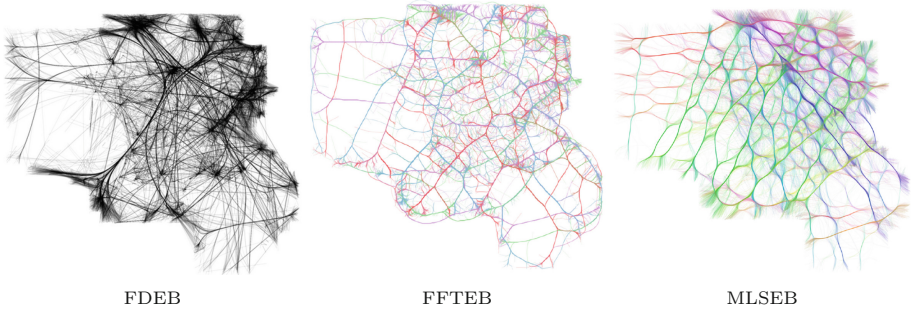


Fig. 4. Visualize the France airlines dataset (17274 edges) with FDEB, FFTEB, and MLSEB.

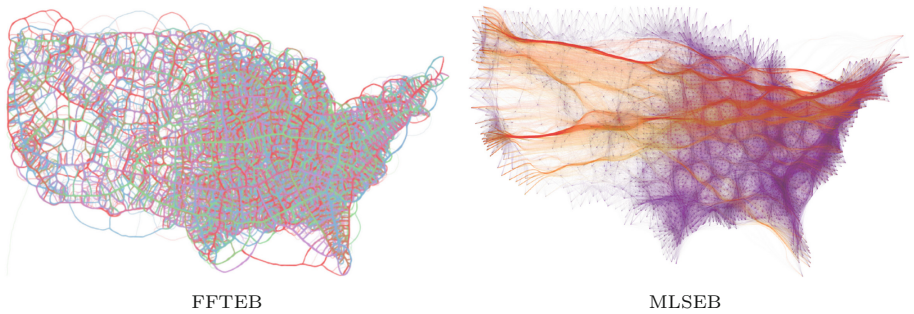


Fig. 5. Comparison of FFTEB and MLSEB using a large US migrations dataset (545881 edges).

Table 1. Performance comparison.

Graph	Edges	FFTEB		MLSEB	
		Samples	Time (ms)	Samples	Time (ms)
US airlines	2180	105 K	40	85 K	22
US migrations	9780	489 K	48	207 K	38
France airlines	17274	864 K	70	990 K	94
Large US migrations	545881	6.4 M	123	5.8 M	554

the reduction of the used pixel number ΔP in a graph drawing to measure C . Specifically, $C = \Delta P = P - P'$ that is the difference of the used pixel number P of the original drawing and the used pixel number P' of the bundled drawing.

Intuitively, T can be given by Eq. 6 that quantifies the total distortion of all the sample points. However, different methods can generate different numbers of sample points. For example, FDEB generates the same number of sample points for each edge, while our MLSEB method and the KDE-based methods

Table 2. Quality comparison using the US migrations graph.

Graph	Edges	FDEB					FFTEB					MLSEB				
		S	P	P'	\bar{T}	Q	S	P	P'	\bar{T}	Q	S	P	P'	\bar{T}	Q
US airlines	2180	813K	32K	25K	1.10K	6.2	105K	32K	18K	1.2K	11.9	85K	32K	19K	0.88K	14.4
US migrations	9780	3785K	34K	26K	0.88K	8.9	489K	32K	24K	1.0K	7.60	207k	33k	25k	0.92k	9.20
France airlines	17274	6685K	81K	72K	2.60K	3.7	864K	81K	57K	1.6K	21.3	990K	81K	60K	0.80K	26.0
Large US migrations	545881	n/a	n/a	n/a	n/a	n/a	6.4M	108k	84k	1.8k	13.3	5.8M	107k	95k	0.90	13.3

sample different edges into different numbers of points. Thus, instead of the total distortion of all the sample points, we use the average distortion: $\bar{T} = \frac{T}{S}$, where S is the total number of the sample points in the graph. Therefore, we modify Eq. 2 to

$$Q = \frac{\Delta P}{\bar{T}}. \quad (11)$$

The rationale of Eq. 11 is to measure how many pixels are decreased by generating one unit distortion. A higher value of Q means a better quality result. Table 2 shows the quantitative quality comparison between our MLSEB method, FDEB and FFTEB. Our comparison is based on the drawings with an image resolution of 400×400 , as shown in Figs. 3, 4 and 5. All the statistic results are generated after a graph is bundled, i.e., after all iterations. We note that it makes less sense to compare the distortion in each iteration because the initial iterations of some methods, such as FDEB and FFTEB, may have surprisingly large distortion. It is more reasonable to compare the quality of results after the bundling iterations are finished. We also note that using different parameters, such as different iteration numbers and different bandwidths for different methods, can yield different results. We use the recommended parameters in FDEB’s and FFTEB’s papers [12, 23], which are the best results we can get from the existing work. The S columns in Table 2 show the numbers of the sample points in a graph using different methods.

We can see that the quality of MLSEB is generally better than the other two methods in terms of Eq. 11. For the four different datasets, FFTEB makes the most clutter reduction. However, it also incurs more distortion. FDEB achieves a comparable quality as ours for the US migrations dataset; whereas, when the dataset is getting larger (France airlines), FDEB will generate tremendous distortion, as shown in Table 2 and Fig. 4, thus lowering the quality score. Note when using the large US migrations dataset, the advantage of MLSEB over FFTEB becomes marginal. Overall, MLSEB gains the highest quantitative scores in terms of quality according to Eq. 11.

7 Conclusions and Future Work

We present a new edge bundling method MLSEB that holistically considers distortion minimization and clutter reduction. Inspired by the MLS work [1, 20], our approach generate bundle effects by iteratively projecting each site to its local regression curve to converge with other nearby sites based on its neighborhood’s

density. Such a local regression curve can reduce the distortion of the local bundle. Our method is easy to implement. The timing result shows MLSEB is at the same order of magnitude of the current fastest edge bundling method FFTEB in terms of computational speed.

We use a quality assessment to evaluate the quality of resulting edge bundles. Our MLSEB method shows better results in our preliminary comparison. However, a more comprehensive comparison between our MLSEB method and the other methods requires further investigation, where other factors (e.g., edge crossing reduction) may be also considered. In addition, we plan to apply optimal bandwidth selection [24, 40] to improve MLSEB. We would also like to incorporate semantic attributes into MLSEB to enhance bundling results. Last but not least, bundling a very large graph (e.g., one with billions or trillions of edges) remains a very challenging task, which is a next possible direction in our future work.

Acknowledgment. This research has been sponsored by the National Science Foundation through grants IIS-1652846, IIS-1423487, and ICER-1541043.

References

1. Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., Silva, C.T.: Computing and rendering point set surfaces. *IEEE Trans. Vis. Comput. Graph.* **9**(1), 3–15 (2003)
2. Bach, B., Riche, N.H., Hurter, C., Marriott, K., Dwyer, T.: Towards unambiguous edge bundling: investigating confluent drawings for network visualization. *IEEE Trans. Vis. Comput. Graph.* **23**(1), 541–550 (2017)
3. Böttger, J., Schäfer, A., Lohmann, G., Villringer, A., Margulies, D.S.: Three-dimensional mean-shift edge bundling for the visualization of functional connectivity in the brain. *IEEE Trans. Vis. Comput. Graph.* **20**(3), 471–480 (2014)
4. Cui, W., Zhou, H., Qu, H., Wong, P.C., Li, X.: Geometry-based edge clustering for graph visualization. *IEEE Trans. Vis. Comput. Graph.* **14**(6), 1277–1284 (2008)
5. Di Battista, G.: *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, Upper Saddle River (1999)
6. Ersoy, O., Hurter, C., Paulovich, F., Cantareiro, G., Telea, A.: Skeleton-based edge bundling for graph visualization. *IEEE Trans. Vis. Comput. Graph.* **17**(12), 2364–2373 (2011)
7. Fröhlich, F., Hross, S., Theis, F.J., Hasenauer, J.: Radial basis function approximations of bayesian parameter posterior densities for uncertainty analysis. In: Mendes, P., Dada, J.O., Smallbone, K. (eds.) *CMSB 2014*. LNCS, vol. 8859, pp. 73–85. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12982-2_6
8. Gansner, E.R., Hu, Y., North, S., Scheidegger, C.: Multilevel agglomerative edge bundling for visualizing large graphs. In: *2011 IEEE Pacific Visualization Symposium*, pp. 187–194. IEEE (2011)
9. Gansner, E.R., Koren, Y.: Improved circular layouts. In: Kaufmann, M., Wagner, D. (eds.) *GD 2006*. LNCS, vol. 4372, pp. 386–398. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70904-6_37
10. Green, S.: Particle simulation using CUDA. *NVIDIA whitepaper* **6**, 121–128 (2010)
11. Holten, D.: Hierarchical edge bundles: visualization of adjacency relations in hierarchical data. *IEEE Trans. Vis. Comput. Graph.* **12**(5), 741–748 (2006)

12. Holten, D., Wijk, J.J.V.: Force-directed edge bundling for graph visualization. *Comput. Graph. Forum* **28**(3), 983–990 (2009)
13. Hurter, C., Ersoy, O., Telea, A.: Graph bundling by kernel density estimation. *Comput. Graph. Forum* **31**(3pt1), 865–874 (2012)
14. Hurter, C., Ersoy, O., Telea, A.: Smooth bundling of large streaming and sequence graphs. In: 2013 IEEE Pacific Visualization Symposium (PacificVis), pp. 41–48. IEEE (2013)
15. Kobourov, S.G., Pupyrev, S., Saket, B.: Are crossings important for drawing large graphs? In: Duncan, C., Symvonis, A. (eds.) GD 2014. LNCS, vol. 8871, pp. 234–245. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45803-7_20
16. Kwon, O.H., Muelder, C., Lee, K., Ma, K.L.: A study of layout, rendering, and interaction methods for immersive graph visualization. *IEEE Trans. Vis. Comput. Graph.* **22**(7), 1802–1815 (2016)
17. Lambert, A., Bourqui, R., Auber, D.: 3D edge bundling for geographical data visualization. In: 2010 14th International Conference Information Visualisation, pp. 329–335, July 2010
18. Lambert, A., Bourqui, R. and Auber, D.: Winding roads: routing edges into bundles. In: *Computer Graphics Forum*, vol. 29, no. 3, pp. 853–862. Wiley (2010)
19. Lancaster, P., Salkauskas, K.: Surfaces generated by moving least squares methods. *Math. Comput.* **37**(155), 141–158 (1981)
20. Lee, I.K.: Curve reconstruction from unorganized points. *Comput. Aided Geom. Des.* **17**(2), 161–177 (2000)
21. Levin, D.: Mesh-independent surface interpolation. In: Brunnett, G., Hamann, B., Müller, H., Linsen, L. (eds.) *Geometric Modeling for Scientific Visualization. Mathematics and Visualization*, pp. 37–49. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-662-07443-5_3
22. Lhuillier, A., Hurter, C., Telea, A.: State of the art in edge and trail bundling techniques. *Comput. Graph. Forum* **36**(3), 619–645 (2017)
23. Lhuillier, A., Hurter, C., Telea, A.: FFTEB: edge bundling of huge graphs by the fast fourier transform. In: PacificVis 2017, 10th IEEE Pacific Visualization Symposium. IEEE (2017)
24. Lipman, Y., Cohen-Or, D., Levin, D.: Error bounds and optimal neighborhoods for MLS approximation. In: *Proceedings of the Fourth Eurographics Symposium on Geometry Processing, SGP 2006*, Eurographics Association, Aire-la-Ville, Switzerland, pp. 71–80 (2006)
25. Luo, S.J., Liu, C.L., Chen, B.Y., Ma, K.L.: Ambiguity-free edge-bundling for interactive graph visualization. *IEEE Trans. Vis. Comput. Graph.* **18**(5), 810–821 (2012)
26. McGee, F., Dingliana, J.: An empirical study on the impact of edge bundling on user comprehension of graphs. In: *Proceedings of the International Working Conference on Advanced Visual Interfaces, AVI 2012*, pp. 620–627. ACM, New York (2012)
27. Mederos, B., Velho, L., Figueiredo, L.H.D.: Moving least squares multiresolution surface approximation. In: *16th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2003)*, pp. 19–26, October 2003
28. Nealen, A.: An As-Short-As-Possible Introduction to the Least Squares, Weighted Least Squares and Moving Least Squares Methods for Scattered Data Approximation and Interpolation (2004)
29. Nguyen, Q., Eades, P., Hong, S.-H.: On the faithfulness of graph visualizations. In: 2013 IEEE Pacific Visualization Symposium (PacificVis), pp. 209–216, February 2013. <https://doi.org/10.1109/PacificVis.2013.6596147>. ISSN:2165-8765

30. Nguyen, Q., Eades, P., Hong, S.-H.: **StreamEB**: stream edge bundling. In: Didimo, W., Patrignani, M. (eds.) GD 2012. LNCS, vol. 7704, pp. 400–413. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36763-2_36
31. Nguyen, Q., Hong, S.-H., Eades, P.: TGI-EB: a new framework for edge bundling integrating topology, geometry and importance. In: van Kreveld, M., Speckmann, B. (eds.) GD 2011. LNCS, vol. 7034, pp. 123–135. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-25878-7_13
32. Paulovich, F.V., Nonato, L.G., Minghim, R., Levkowitz, H.: Least square projection: a fast high-precision multidimensional projection technique and its application to document mapping. *IEEE Trans. Vis. Comput. Graph.* **14**(3), 564–575 (2008)
33. Peysakhovich, V., Hurter, C., Telea, A.: Attribute-driven edge bundling for general graphs with applications in trail analysis. In: 2015 IEEE Pacific Visualization Symposium (PacificVis), pp. 39–46. IEEE (2015)
34. Pupyrev, S., Nachmanson, L., Kaufmann, M.: Improving layered graph layouts with edge bundling. In: Brandes, U., Cornelsen, S. (eds.) GD 2010. LNCS, vol. 6502, pp. 329–340. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-18469-7_30
35. Purchase, H.: Which aesthetic has the greatest effect on human understanding? In: DiBattista, G. (ed.) GD 1997. LNCS, vol. 1353, pp. 248–261. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63938-1_67
36. Purchase, H.C., Cohen, R.F., James, M.: Validating graph drawing aesthetics. In: Brandenburg, F.J. (ed.) GD 1995. LNCS, vol. 1027, pp. 435–446. Springer, Heidelberg (1996). <https://doi.org/10.1007/BFb0021827>
37. Selassie, D., Heller, B., Heer, J.: Divided edge bundling for directional network data. *IEEE Trans. Vis. Comput. Graph.* **17**(12), 2354–2363 (2011)
38. Tamassia, R.: *Handbook of Graph Drawing and Visualization (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC (2007)
39. Telea, A., Ersoy, O., Hoogendorp, H., Reniers, D.: Comparison of node-link and hierarchical edge bundling layouts: a user study. In: Keim, D.A., Pras, A., Schönwälder, J., Wong, P.C. (eds.) *Visualization and Monitoring of Network Traffic*. No. 09211 in Dagstuhl Seminar Proceedings, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, Dagstuhl, Germany (2009)
40. Wang, H., Scheidegger, C.E., Silva, C.T.: Bandwidth selection and reconstruction quality in point-based surfaces. *IEEE Trans. Vis. Comput. Graph.* **15**(4), 572–582 (2009)
41. Wu, J., Yu, L., Yu, H.: Texture-based edge bundling: a web-based approach for interactively visualizing large graphs. In: 2015 IEEE International Conference on Big Data (Big Data), pp. 2501–2508, October 2015
42. Zhou, H.: *Visual Clustering in Parallel Coordinates and Graphs*. Ph.D. thesis (2009), aAI3398258
43. Zielasko, D., Weyers, B., Hentschel, B., Kuhlen, T.W.: Interactive 3D force-directed edge bundling. In: *Computer Graphics Forum*, vol. 35, no. 3, pp. 51–60. Wiley (2016)
44. van der Zwan, M., Codreanu, V., Telea, A.: CUBu: universal real-time bundling for large graphs. *IEEE Trans. Vis. Comput. Graph.* **22**(12), 2550–2563 (2016)

Drawing Dynamic Graphs Without Timeslices

Paolo Simonetto¹, Daniel Archambault^{1(✉)}, and Stephen Kobourov²

¹ Swansea University, Swansea, UK

{paolo.simonetto,d.w.archambault}@swansea.ac.uk

² University of Arizona, Tucson, USA

kobourov@cs.arizona.edu

Abstract. Timeslices are often used to draw and visualize dynamic graphs. While timeslices are a natural way to think about dynamic graphs, they are routinely imposed on continuous data. Often, it is unclear how many timeslices to select: too few timeslices can miss temporal features such as causality or even graph structure while too many timeslices slows the drawing computation. We present a model for dynamic graphs which is not based on timeslices, and a dynamic graph drawing algorithm, DynNoSlice, to draw graphs in this model. In our evaluation, we demonstrate the advantages of this approach over timeslicing on continuous data sets.

1 Introduction

Graphs offer a natural way to represent a static set of relations. In order to encode changes to a graph over time, static graphs can be extended to dynamic graphs. Dynamic graphs are traditionally thought of as a sequence of static graphs in a finite number of moments in time, or *timeslices*. We refer to these dynamic graphs as *discrete dynamic graphs*. Discrete dynamic graphs are widely used for several reasons. Drawing timesliced graphs is similar to drawing static graphs, allowing force-directed approaches to be easily adapted to dynamic data. Also, timeslicing works fine for data that changes at discrete, regular time intervals.

Algorithms to draw discrete dynamic graphs strike a balance between graph drawing readability and stability [3]. Readability requires the drawing of individual timeslices to be of high quality while stability (mental map preservation) requires nodes to not move too much between consecutive timeslices for easy identification [2]. These requirements conflict with each other as node movement is required for timeslice readability, while it negatively impacts stability.

Approaches for discrete dynamic graph drawing extend static graph drawing algorithms. Each timeslice is a static graph that is drawn considering the timeslices before and after to stabilize the transition. However, when the time dimension is continuous, there are no timeslices. Thus, evenly spaced timeslices are selected and the graph elements are projected to the closest one. This operation implicitly aggregates data along the time dimension (Fig. 1). Selecting too many timeslices leads to slow layout computation while selecting too few leads to information loss due to projection errors when events are of short duration,

obscuring graph structure. Even for graphs with nodes and edges that span long periods of time, regular timeslices can obscure important temporal patterns.

A natural solution to the problems above would be to refrain from imposing timeslices on the continuous data and draw the graph directly along a continuous time dimension. In a recent state-of-the-art report, Beck *et al.* [7, p. 15] states “the effects of using continuous time with arbitrary fine sampling rates, rather than discretized time, are largely unexplored”. In this paper, we introduce a model for dynamic graph drawing that does not use timeslices. We call these graphs *continuous dynamic graphs* and propose the first dynamic graph drawing algorithm, DynNoSlice, to draw them along a continuous time dimension.

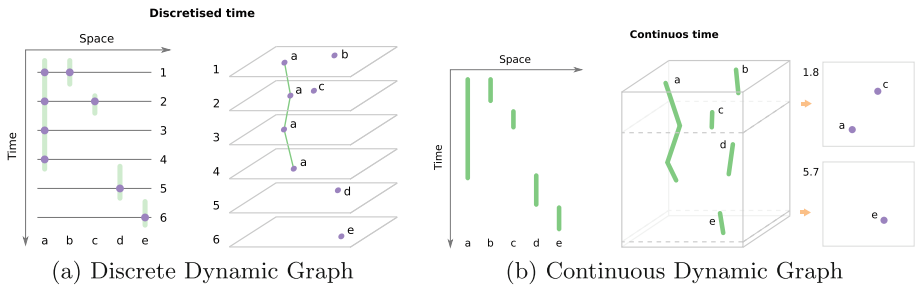


Fig. 1. A dynamic graph with 5 nodes. (a) A discrete representation with nodes projected onto timeslices. Projecting the nodes (purple dots) results in information loss due to aggregation across time. (b) In our approach, nodes are defined as piecewise linear curves in the space-time cube, as timeslices are not imposed on continuous data. (Color figure online)

2 Related Work

Dynamic graphs visualization is a well established field of research, as shown in the recent survey by Beck *et al.* [7]. If we were to place our approach in this model, it would be an “offline time-to-time mapping” without timeslices, as it works with full knowledge of the input data. Related approaches also include the “superimposed time-to-space mapping”, as this approach inherently defines a space-time cube.

Offline Drawing. Offline dynamic graph drawing algorithms capture all of the dynamic data beforehand and optimize across it simultaneously. Foresighted layout algorithms [17, 18] are among the first attempts at offline dynamic graph drawing. They create a supergraph as the union of the elements at each timeslice. The supergraph is then used to define the position of nodes and edge bends. Mental map preservation is the primary focus of this approach.

Brandes and Corman [8] designed an approach for offline dynamic graph drawing that visually refers to time as a third, discrete spacial component. Timeslices are placed on top of each other, forming a layered space-time cube.

A similar technique is also used by Dwyer and Eades [19]. In both approaches, the time dimension is modeled using timeslices.

Erten *et al.* [21] explored several ways to superimpose different graphs, including the use of different colors for different timeslices of a layered space-time cube. In GraphAEL [20,23], the space-time cube was created by connecting the same nodes in consecutive timeslices with inter-timeslice edges and optimizing this structure with a force-directed algorithm. A similar technique is used by Groh *et al.* [30] and by Itoh *et al.* [32] for social networks. Brandes and Mader [9] perform a metric-based evaluation of several strategies for drawing discrete dynamic graphs including aggregation, anchoring, and linking. Linking strategies performed best in balancing drawing quality (measured using stress) and mental map preservation (measured using distance traveled). Rauber *et al.* [43], identify vectors $v_i[t]$ in the cost gradient whose geometrical interpretation would be identical when working in a layered space-time cube.

Our approach can be seen as an extension of offline discrete dynamic graph drawing approaches to continuous time. However, the difference is substantial as in related work timeslices are imposed on the data. By drawing in continuous time, each node and edge defines its own trajectory with its own complexity whereas discrete algorithms impose linear interpolations between timeslices.

Time-to-space mapping approaches [7] use one dimension for space and one dimension for time in order to visualize the network [12,36,45,49]. Recent scalable approaches use dimensionality reduction to show time as a curve in the plane [6,50]. By using spatial dimensions to represent time, these visualizations are substantially different from classic dynamic node-link diagrams.

Online Drawing. Given an incoming stream of data, online drawing algorithms continuously update the current graph drawing to take into account new data. Misue *et al.* [37] optimize mental map preservation by maintaining the horizontal and vertical order of the nodes through adjustments. Frishman and Tal [25] proposed a different force-directed algorithm which could also be partially executed on the GPU. Goroehowski *et al.* [28] use node aging to decide how much a node position should be preserved at a given time. Finally, Crnovrsanin *et al.* [14] adapted the FM³ [31] multilevel approach to an online setting.

Although related, online approaches are different as they operate under more stringent constraints due to lack of access to future information. Offline algorithms, such as DynNoSlice, use the full knowledge of the graph evolution.

3D Graph Drawing. Several 2D algorithms have been extended to 3D. Bruß and Frick [11] proposed Gem3D, which extends Gem [24] to 3D. Cruz and Twarog [15] extended the simulated annealing approach of Davidson and Harel [16] to 3D. Other algorithms work in both 2D and 3D, such as GRIP by Gajer and Kobourov [26]. Munzner [38] proposed an algorithm to draw directed graphs in a 3D hyperbolic space. Several other algorithms deal with particular constraints, such as orthogonal [34], or nested [41] drawings. Cordeil *et al.* [13] investigate the visualization of graphs in 3D using immersive environments.

Although our approach does compute a 3D layout of the dynamic graph, we do not produce a 3D visualization. Since our third dimension is time, nodes and edges have additional constraints and are not free to move arbitrarily in 3D.

Space-Time Cubes. Sallaberry *et al.* [44] visualize dynamic graphs by manipulating the space-time cube. Several other information visualization methods perform similar operations in the space-time cube, as discussed in Bach *et al.* [5]. These approaches assume that the space-time cube is already given and focus on how to accommodate it in 2D, whereas our approach constructs the space-time cube for dynamic graphs using a continuous time dimension.

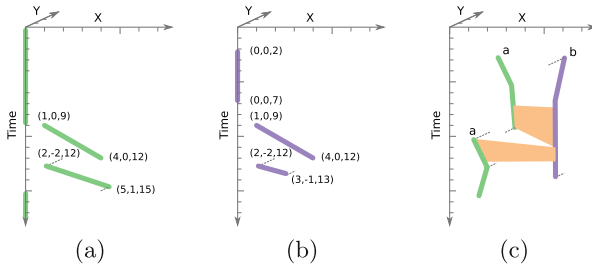


Fig. 2. A continuous dynamic graph in the space-time cube. (a) Position attribute for a node v . (b) The appearance attribute. (c) An edge in a continuous dynamic graph, as encoded by the edge appearance attribute.

3 Continuous Dynamic Graph Model

Let $G = (V, E)$ be a static graph defined with node set V and edge set $E \subseteq V \times V$. We define *attributes* (functions) on the nodes and edges of the graph that encode characteristics such as their positions, weights, and labels. The node position attribute ($\mathcal{P}_G : V \rightarrow \mathbb{R}^2$) maps a node v into its position in the 2D plane \mathbf{p}_v , e.g., $\mathcal{P}_G(v) = (1, 4)$. This attribute is not integral to our model of continuous dynamic graphs, but is used to compute and store the layout of these graphs.

We define a continuous dynamic graph $D = (V, E)$ as a graph whose attributes are also a function of time. Let T be the time domain defined as an interval in \mathbb{R} . Attributes are functions defined in the domain $V \times T$ for nodes, and $E \times T$ for edges. For example, the node position attribute $\mathcal{P}_D : V \times T \rightarrow \mathbb{R}^2$ is the function that describes the position of v for each time $t \in T$. We assume (w.l.o.g.) that the node and edge attributes can be defined piecewise, e.g.:

$$\mathcal{P}_D(v, t) = \mathcal{P}_v(t) = \begin{cases} \mathcal{P}_{v,1}(t) & \text{for } t \in T_1 \\ \vdots & \\ \mathcal{P}_{v,n}(t) & \text{for } t \in T_n \\ \mathbf{p}_{v,\omega} & \text{otherwise} \end{cases}$$

In other words, a dynamic attribute can be thought of as a map that links each node (edge) to a sequence of functions $\mathcal{P}_{v,i}$ that describe its behavior in disjoint intervals of time T_i , with a default value returned for $t \notin \bigcup_i T_i$. In the rest of paper we consider only piecewise linear functions for nodes and edges.

Attributes specify a variety of node and edge characteristics. For data that can be meaningfully interpolated (e.g., colors, weights, positions), the functions above can be described by initial and final values. For attributes without meaningful interpolation (e.g., labels), we prefer functions that are constant in the related interval. Position and label attributes for a node v of D can be:

$$\mathcal{P}_v(t) = \begin{cases} (1, 0) \rightarrow (4, 0) & \text{for } t \in (9, 12] \\ (2, -2) \rightarrow (5, 1) & \text{for } t \in (12, 15] \\ (5, 1) \rightarrow (4, 5) & \text{for } t \in [17; 19] \\ (0, 0) & \text{otherwise} \end{cases} \quad \mathcal{L}_v(t) = \begin{cases} \text{Jane Doe} & \text{for } t \in (10, 11] \\ \text{Jane Smith} & \text{for } t \in (11, 16] \\ \text{unknown} & \text{otherwise} \end{cases}$$

We define the attribute *appearance* that implements the classic dynamic graph operations node/edge insertion and deletion.

$$\mathcal{A}_v(t) = \begin{cases} \text{true} & \text{for } t \in [2, 7) \\ \text{true} & \text{for } t \in (9, 13] \\ \text{false} & \text{otherwise} \end{cases}$$

In order to mark an edge $e = (u, v)$ as present in T_x , we need to ensure that both u and v are present for the entire T_x .

This definition supports changes in node (edge) characteristics at any time, whereas discrete dynamic graphs allow changes only to occur in timeslices. DynNoSlice implements the above model as a collection of piecewise, linear functions defined on intervals in the space-time cube. Fast access to these functions at any given time is required. In our implementation, we use interval trees. When the intervals are guaranteed to be non-overlapping, simpler structures such as binary search trees can be used.

4 DynNoSlice Implementation

A continuous dynamic graph D can be transformed into a 3D static graph D' by embedding it in the space-time cube. Algorithms for static or discrete dynamic graphs can be extended to work with this new representation. In this section, we describe our force-directed algorithm for drawing continuous dynamic graphs. It has been implemented and the source code is available¹.

¹ <http://cs.swan.ac.uk/~dynnoslice/software.html>.

4.1 Representation in the Space-Time Cube

We can define a space-time cube transformation (STCT) that transforms a continuous dynamic graph into a drawing in the space-time cube. In D' , the presence and position of each node is represented by a sequence of trajectories. The shapes of these trajectories are defined by the position attributes. In the example above, the trajectory of v is defined by three line segments, $(1, 0, 9) \rightarrow (4, 0, 12)$, $(2, -2, 12) \rightarrow (5, 1, 15)$ and $(5, 1, 17) \rightarrow (4, 5, 19)$, and by portions of the line $(0, 0, x)$ (see Fig. 2a). The number of these trajectories is also affected by the appearance attribute. In the example above, the node v appears two times: at $[2, 7)$ and at $(9, 13]$. Clearly, the behavior of the node at times when it is not part of the graph is non-influential. Therefore, the node appearance and position in the space-time cube can be identified by the segments $s_{v,1} = (0, 0, 2) \rightarrow (0, 0, 7)$, $s_{v,2} = (1, 0, 9) \rightarrow (4, 0, 12)$ and $s_{v,3} = (2, -2, 12) \rightarrow (3, -1, 13)$ (see Fig. 2b). The trajectory given by the polyline is made of the start and end points, as well as the bends (the junctions of consecutive segments).

The representation of edges in the space-time cube is less intuitive. By connecting two trajectories with lines in the space-time cube, we obtain a ruled surface. Therefore, an edge $e = (u, v)$ is a surface that connects trajectories u and v for a duration indicated by \mathcal{A}_e . If the node trajectories are not continuous as in the above example:

$$\lim_{t \rightarrow 12^-} \mathcal{P}_v(t) = (4, 0) \quad \text{and} \quad \lim_{t \rightarrow 12^+} \mathcal{P}_v(t) = (2, -2),$$

an edge might create two or more surfaces (see Fig. 2c).

If the trajectory segments, considered as vectors, form an acute angle with respect to the positive time axis, this transformation is easily invertible. Thus, trajectory segments cannot fold back on themselves, as it would identify multiple positions for that node at a given time. We can then work on this continuous dynamic graph in 3D as if it were a static graph as follows:

$$D_a \rightarrow \text{STCT} \rightarrow D'_a \rightarrow \text{Operation} \rightarrow D'_b \rightarrow \text{STCT}^{-1} \rightarrow D_b$$

4.2 Force-Directed Drawing Algorithm

Our force-directed algorithm is based on earlier variants by Simonetto *et al.* [46,47]. As in most force-directed algorithms, after an initialization phase, the algorithm iteratively improves the current layout of the drawing in 3D for a given number of iterations in the following way:

- For each iteration of the algorithm:
 - Compute and sum the forces based on the force system.
 - Move nodes based on these forces and the constraints.
 - Adjust trajectory complexity in the space-time cube.

Initialization. Each node v is randomly assigned a position (a, b) in the 2D plane. The points defining v 's trajectory are extruded linearly along the time axis (a, b, x) , where x is the time coordinate.

Forces. DynNoSlice has five forces. The first three forces adapt standard, force-directed approaches to work with trajectories in the space-time cube. The final two are novel for continuous dynamic graph drawing. In our notation, a star transforms 3D vector to 2D by dropping the time coordinate. For example, if $p = (1, 2, 3)$ then p^* is $(1, 2)$. The parameter δ is an ideal distance between two node trajectories.

1. *Node Repulsion.* This force repels trajectories from each other. The force evenly distributes node trajectories in space and prevents crowding [35, 42].

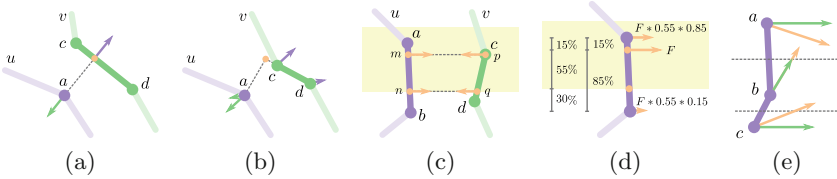


Fig. 3. Forces and constraints. (a) and (b) Node repulsion force between point a and the line segment $c \rightarrow d$ that represents the trajectories of nodes u and v . (c) and (d) Edge attraction for an edge in the interval highlighted with yellow background. (e) Time movement restriction. Endpoints (a and c) must keep their assigned time coordinates. Bend b cannot move past half the distance with other bends or endpoints.

For each segment endpoint a in the trajectory of node u and segment $s_{v,j} = c \rightarrow d$ of node v , with $u \neq v$, we compute the forces generated on the points a , c and d (see Fig. 3). If the points a , c and d are not collinear and $(p \in s_{v,j})$, they form a plane. In this case, we apply the force *EdgeNodeRepulsion* (δ) described in previous work [47], except node positions are in 3D. If the points are collinear or the projection p of a does not fall in the segment $s_{v,j}$ ($p \notin s_{v,j}$), we apply *NodeNodeRepulsion*(δ) [47] between a and c and between a and d .

Since distant segments do not interact significantly, they can be ignored to reduce the running time. A multi-level interval tree is used to identify segments that are sufficiently close ($<5\delta$). All other pairs are ignored.

2. *Edge Attraction.* This attractive force pulls trajectories that are linked by an edge closer to each other. The force is exerted only for the intervals where the edge is present.

Let us consider an edge $e = (u, v)$ that appears at interval (t_1, t_2) and let $I = (t_1\tau, t_2\tau)$ be the transformed time interval in the space-time cube with conversion factor τ that transforms time into the third dimension of the space time cube. For each pair of segments $s_{u,i} = a \rightarrow b$ and $s_{v,j} = c \rightarrow d$ that overlap with the interval (f, g) , we compute the points m, n of segment $s_{u,i}$ and p, q of segment $s_{v,j}$ so that:

$$\min \{f \in I : \exists m \in s_{u,i}, \exists p \in s_{v,j}, f = m[2] = p[2]\}$$

$$\max \{g \in I : \exists n \in s_{u,i}, \exists q \in s_{v,j}, g = n[2] = q[2]\}$$

where $x[2]$ is the time coordinate of the point x in the space-time cube. We compute the attractive force between the points m and p , and n and q , using EdgeContraction (δ) [47] (see Fig. 3c). This force is applied to the segment endpoints once scaled by its distance from the application point and by the coverage of the edge appearance on the segment (see Fig. 3d). For example, if the force F_m attracts m to p , then F_a applied to a will be:

$$F_a = F_m * \frac{a[2] - m[2]}{a[2] - b[2]} * \frac{n[2] - m[2]}{a[2] - b[2]}.$$

3. *Gravity.* This force encourages a compact drawing of node trajectories. Let c be the center in 2D of the initial node placement in the space-time cube. The gravity F of each segment endpoint a is $F = c^* - a^*$.

4. *Trajectory Straightening.* This force smooths node trajectories, helping with node movements over time. For trajectory bends, we use the CurveSmoothing [47] force, which pulls a bend b to the centroid of the triangle Δabc formed by using the previous and next bends or endpoints a and c . A trajectory endpoint a has no such triangle. Therefore, it is pulled in 2D toward the midpoint of the segment formed with the closest bend or endpoint b .

5. *Mental Map Preservation.* This force prevents trajectory segments from making large angles with respect to time. When segments form a 90° angle with time, a node essentially “teleports” from one place to another, while segments parallel to the time axis result in no node movement. Thus, segments should form small angles with the time axis. This force pulls endpoints a and b of each trajectory towards each other in 2D with a magnitude based on the angle α the segment makes with the time axis:

$$F_a = (b^* - a^*) * \frac{\alpha}{90^\circ - \alpha}$$

Constraints. Node movement constraints ensure valid drawing of the continuous dynamic graph in the space-time cube. In particular, constraints are needed to prevent undesired movements.

Decreasing Max Movement. We insert a constraint on the maximum node movement allowed at each iteration. This allows large movements at the beginning of the computation, and smaller refinements towards the end. This constraint is similar to DecreasingMaxMovement (δ) [47].

Movement Acceleration. This constraint promotes consistent movements with previous iterations and penalizes movements in the opposite direction. This constraint corresponds to MovementAcceleration (δ) [47].

Time Correctness. This constraint prevents a node from changing its time coordinate. Consider the trajectory formed by the segment $t = a \rightarrow b$. If a changes its time coordinate in the space-time cube, the time of its appearance will also change. Now, consider a trajectory formed by several segments, $t = a \rightarrow b \rightarrow c \rightarrow d$. The trajectory endpoints a and d , corresponding to the

appearance and disappearance of the node, should have fixed time positions. However, bends b and c can move in time, so long as they do not pass each other ($a[2] < b[2] < c[2] < d[2]$) as this would result in a node being in two locations at once. Therefore, the movement M_a of node a in time is constrained to be: $M_a \leftarrow M_a^*$ if a is the endpoint of a trajectory, and

$$M_b \leftarrow M_b * \max \left\{ r \in [0, 1] : \frac{M_a[2] - M_b[2]}{2} < rM_b[2] < \frac{M_c[2] - M_b[2]}{2} \right\}$$

if b is a trajectory bend between other bends or endpoints a and c (see Fig. 3e).

Complexity Adjustment of Node Trajectories. As bends move freely in time and space, node trajectories can be oversampled or undersampled. Therefore, bends can be inserted or removed from the polyline representing a node trajectory. If a segment of the trajectory between bends a and b is greater than a threshold (2δ in this paper), a bend is inserted at its midpoint. Similarly, if two consecutive segments ab and bc are placed such that the distance between a and c is less than a threshold (1.5δ in this paper), the bend b is removed and the two segments are replaced by ac .

5 Evaluation

We perform a metric-based evaluation of our approach against the state-of-the-art algorithm `visone` [10] to demonstrate the advantages of our model.

5.1 Data Sets

InfoVis Co-Authorship (Discrete): a co-authorship network for papers published in the InfoVis conference from 1995 to 2015 [1]. Authors collaborating on a paper are connected in a clique at the time of publication of the paper. Note this is not a cumulative network as authors can appear, disappear, and appear again. The data is of discrete nature with exactly 21 timeslices (one per year).

Van De Bunt (Discrete): shows the relationships between 32 freshmen at seven different time points. A discrete dynamic graph is built using the method of Brandes *et al.* [9], with an undirected edge inserted into a timeslice if the participants reciprocally report “best friendship” or “friendship” at that time.

Newcomb Fraternity Data (Discrete): contains the sociometric preference of 17 members of a fraternity in the University of Michigan in the fall of 1956 [39]. As in previous work [9], at each timeslice, we inserted undirected edges connecting students to their three best friends.

Rugby (Continuous): is a network derived from over 3,000 tweets involving teams in the Guinness Pro12 rugby competition. The tweets were posted between 09.01.2014 and 10.23.2015. Each tweet contains information about the involved teams and the time of publication with a precision down to the second.

Pride and Prejudice (Continuous): lists the dialogues between characters in the novel *Pride and Prejudice* in order [29]. The book has 61 chapters and the data set includes over 4,000 interactions between characters.

5.2 Method

As there are no continuous dynamic graph drawing algorithms, we need to compare our results with discrete dynamic graph drawing algorithms. In our metric evaluation, we considered three drawing approaches:

- Visone (v) drawings were computed using the discrete version (timesliced) with `visone` [10]. We use a linking strategy [9] with a default link length of 200 and a stability parameter $\alpha = 0.5$.
- Discrete (d) drawings were computed using a modified version of `DynNoSlice` on the discrete version. Each trajectory bend coincides with a timeslice and bends cannot be inserted or removed. The rest of the algorithm is unaltered. We used $\delta = 1$ as the desired edge length parameter.
- Continuous (c) drawings were computed using `DynNoSlice`. We used $\delta = 1$ as the desired edge length parameter.

We evaluate the results using a variety of metrics:

- Stress: the average edge stress, as defined by Brandes and Mader [9, Formula 1]. As stress is defined for a static graph, we slice the space-time cube and average the stress computed on each timeslice.
- Node Movement: the average 2D movements of the nodes. Intuitively, this is the average distance traveled by nodes when animating the dynamic graph.
- Crowding: This metric counts the number of times nodes collide during the animation of the dynamic graph.
- Running Time in seconds.

For continuous dynamic graphs, we can compute the stress of the nodes and edges present in the timeslice defined by the discrete dynamic graph or on the node and edge set at that precise point in continuous time. We can also compute stress between timeslices. Thus, we consider four measures of stress:

- StressOn (d): the stress computed on the timeslices using the node and edge set of that timeslice.
- StressOff (d): the stress computed on and between the default timeslices using the node and edge set of the closest timeslice in time, when between two timeslices.
- StressOn (c): the stress computed on the timeslices using the precise node and edge appearances in continuous time.
- StressOff (c): the stress computed on and between the timeslices using the precise node and edge appearances in continuous time.

Graph Scaling. Uniformly scaling node positions changes the measure of stress even though the layout is the same [27, 33, 40]. In order to compare methods as fairly as possible, we used a strategy where scale-independent values of stress are compared as follows. Both `visone` and `DynNoSlice` have a parameter that indicates the desired edge length. First, we verified that `visone` produces the

same result (up to scale) when changing the edge length parameter. Thus, we use the default value of edge length but consider different scaling factors to compare to the output of our algorithm. For the experiment, we defined nodes to be circles of diameter 0.2 and with an ideal edge length of 1. To obtain such drawing with our approach, we run the algorithm with an ideal edge length parameter $\delta = 1$. To obtain such drawing with **visone**, we run it with the default edge length of 200 and scale it down by a factor 200.

Related work in static graph drawing [27, 33, 40] searches for the best scaling factor via binary search, as a minimum is guaranteed. For our metric (average stress across all timeslices) we have no such guarantee. Thus, we evaluate scaling factors $(1.1)^i$, with $i \in \mathbb{Z} : -20 < i < 20$ for the best StressOn(d) value. This scaling factor is used to compute all metrics. After plotting the average stress for each data set, a minimum in this range was consistently observed.

5.3 Results

Videos of *Newcomb*, *Rugby*, and *Pride and Prejudice* are available². In this section, we present our quantitative results.

visone is often faster than our continuous approach as DynNoSlice operates on a greater volume of data – the data between timeslices. The discrete version of DynNoSlice is often slower than both, as the continuous approach is more naturally expressed without timeslices. Therefore, there is a penalty for imposing timeslices on it.

Table 1 shows the results on the discrete data sets. Continuous stress metrics are not computed, because nodes and edges only appear on timeslices. In the VanDeBunt and Newcomb data sets, **visone** outperforms DynNoSlice in terms of stress. Movement and crowding are comparable among all three approaches. Our continuous approach is sometimes able to improve on **visone** when stress is computed off timeslices. When comparing the discrete and continuous versions of DynNoSlice, our discrete version is often able to optimize on-timeslice stress.

Table 1. Results for the discrete data sets on our metrics.

Graph	Type	Time (s)	Scale	StressOn (d)	StressOff (d)	Movement	Crowding
VanDeBunt	v	0.13	1.00	1.14	1.46	3.80	0
	d	7.73	0.62	1.20	1.20	3.91	0
	c	6.73	0.68	1.19	1.29	3.69	0
Newcomb	v	0.11	1.00	14.04	14.77	16.36	8
	d	9.68	0.68	16.61	16.59	13.48	2
	c	7.62	0.75	18.13	17.98	12.37	0
InfoVis	v	77.43	0.47	51.66	52.98	2.15	36
	d	388.38	0.56	31.09	31.04	2.03	8
	c	381.15	0.56	32.70	34.02	1.91	6

² <http://cs.swan.ac.uk/~dynnoslice/files/video.mp4>.

InfoVis is an outlier for the timesliced data sets. In particular, our continuous approach outperforms **visone** in terms of stress.

Table 2 shows the results of our metric experiment on the continuous data sets. Our continuous approach has lower off-timeslice stress, lower average movement, fewer crowding events, and occasionally lower on-timeslice stress.

Table 2. Results for the continuous data sets on our metrics.

Graph	Type	Time (s)	Scale	StressOn (c)	StressOff (c)	Movement	Crowding
Rugby	v	0.08	0.68	3.08	2.71	25.47	6
	d	7.40	0.68	1.84	1.71	16.23	1
	c	3.88	0.51	1.84	1.77	6.57	0
Pride	v	3.39	0.18	0.62	0.88	5.44	682
	d	1655.50	0.32	0.82	0.86	6.95	13
	c	75.61	0.24	0.83	0.85	1.12	3

5.4 Discussion

Our results on the discrete data sets were expected: **visone** optimizes for stress directly on every timeslice and so outperforms our force system in terms of stress, while it is comparable in terms of movement and crowding. As a state-of-the-art algorithm for timesliced graph drawing, it is difficult to compete with **visone** when it is running on the type of data for which it was designed. However, when stress is measured off the timeslices, our continuous approach often outperforms **visone**. The timesliced model does not allow for stress to be optimized between timeslices and must resort to linear interpolation, leading to suboptimal stress. In our continuous dynamic graph model, we optimize for stress in continuous time, leading to this performance improvement. The InfoVis data set is an exception where DynNoSlice is able to improve on **visone** in terms of stress. This result could be due to the bursty nature of this graph (edges are only present if two authors published a joint paper that year). Therefore, large parts of the graph change drastically from year to year. Allowing node trajectories to evolve independently of timeslices may allow DynNoSlice to perform better.

For nearly all continuous data sets, our continuous model often outperforms **visone** in terms of stress, movement, and crowding. This is likely due to the fact we do not use timeslices to compute the layout of the dynamic graph. As a result, we are able to optimize stress between timeslices as well. On-timeslice stress is an exception, as it is directly optimized by **visone**.

It is surprising that our continuous dynamic graph drawing algorithm simultaneously improves node movement and crowding while remaining competitive or improving on stress. This finding may seem counter-intuitive as low stress usually corresponds to high node movement. This result can be explained by the fact that nodes in our continuous models are polylines of adaptive complexity

in the space-time cube. Nodes with few interactions in the data will be long straight lines, potentially passing through many timeslices. These areas of low complexity will reduce average node movement. In a model that uses timeslices, each timeslice is forced to have that node with inter-timeslice edges. Therefore, timeslices impose additional node movement that may not be necessary.

In terms of crowding, the continuous model allows the polyline representing a node to adapt its complexity between timeslices if there are many interactions. When there are many changes to the graph in a short period of time, these polylines have increased complexity, allowing nodes to avoid crowding. In a timesliced model, only linear interpolation is possible, and all nodes must follow straight lines. Thus, crowding is incurred. Crowding is also avoided in our continuous model as our polylines have repulsive forces between them. In all timeslice approaches, inter-timeslice edges do not repel each other, potentially causing crowding events.

6 Conclusions and Future Work

We presented a model for dynamic graph drawing without timeslices in which nodes and edges, along with their attributes, are defined on continuous time intervals. We developed a dynamic graph drawing algorithm, DynNoSlice, that visualizes graphs in this model by working on the space-time cube. An implementation of this algorithm is available along with a video. In our evaluation, we demonstrate that our continuous approach has significant advantages over timeslicing the continuous data.

The focus of this paper is a method to draw dynamic graphs without timeslices. An animation of a slice traversing the layout in the space-time cube is a natural visualization. However, animation is not always effective in terms of human performance [4, 22, 48], especially when events are of short duration. More effective visual representations the layout present in the space-time cube are necessary.

The primary issue that we have found with collapsing continuous time down onto a series of timeslices is that the timeslices could oversample/undersample the data in the continuous dimension. In signal processing, the Nyquist frequency gives the minimum sampling rate required to reconstruct the signal. Regular timeslices taken at the smallest temporal distance between two events should be sufficient to avoid undersampling, but lower sampling frequencies could be possible and remain future work.

References

1. Citevis citation datafile. <http://www.cc.gatech.edu/gvu/ii/citevis/infocitation-data.txt>. Accessed 28 Aug 2017
2. Archambault, D., Purchase, H.C.: Mental map preservation helps user orientation in dynamic graphs. In: Didimo, W., Patrignani, M. (eds.) GD 2012. LNCS, vol. 7704, pp. 475–486. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36763-2_42

3. Archambault, D., Purchase, H.C.: Can animation support the visualization of dynamic graphs? *Inf. Sci.* **330**, 495–509 (2016)
4. Archambault, D., Purchase, H.C., Pinaud, B.: Animation, small multiples, and the effect of mental map preservation in dynamic graphs. *IEEE Trans. Vis. Comput. Graph.* **17**(4), 539–552 (2011)
5. Bach, B., Dragicevic, P., Archambault, D., Hurter, C., Carpendale, S.: A descriptive framework for temporal data visualizations based on generalized space-time cubes. In: *Computer Graphics Forum* (2016)
6. Bach, B., Shi, C., Heulot, N., Madhyastha, T., Grabowski, T., Dragicevic, P.: Time curves: folding time to visualize patterns of temporal evolution in data. *IEEE Trans. Vis. Comput. Graphics (InfoVis 2015)* **22**(1), 559–568 (2016)
7. Beck, F., Burch, M., Diehl, S., Weiskopf, D.: The state of the art in visualizing dynamic graphs. In: Borgo, R., Maciejewski, R., Viola, I. (eds.) *EuroVis - STARSS*. The Eurographics Association (2014)
8. Brandes, U., Corman, S.R.: Visual unrolling of network evolution and the analysis of dynamic discourse. *Inform. Vis.* **2**(1), 40–50 (2003)
9. Brandes, U., Mader, M.: A quantitative comparison of stress-minimization approaches for offline dynamic graph drawing. In: van Kreveld, M., Speckmann, B. (eds.) *GD 2011*. LNCS, vol. 7034, pp. 99–110. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-25878-7_11
10. Brandes, U., Wagner, D.: Analysis and visualization of social networks. In: Jünger, M., Mutzel, P. (eds.) *Graph Drawing Software*. Mathematics and Visualization, pp. 321–340. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-642-18638-7_15
11. Bruß, I., Frick, A.: Fast interactive 3-D graph visualization. In: Brandenburg, F.J. (ed.) *GD 1995*. LNCS, vol. 1027, pp. 99–110. Springer, Heidelberg (1996). <https://doi.org/10.1007/BFb0021794>
12. Burch, M., Vehlow, C., Beck, F., Diehl, S., Weiskopf, D.: Parallel edge splatting for scalable dynamic graph visualization. *IEEE Trans. Vis. Comput. Graphics (InfoVis 2011)* **17**(12), 2344–2353 (2011)
13. Cordeil, M., Dwyer, T., Klein, K., Laha, B., Marriott, K., Thomas, B.H.: Immersive collaborative analysis of network connectivity: CAVE-style or head-mounted display? *IEEE Trans. Vis. Comput. Graphics (InfoVis 2016)* **23**(1), 441–450 (2017)
14. Crnovrsanin, T., Chu, J., Ma, K.-L.: An incremental layout method for visualizing online dynamic graphs. In: Di Giacomo, E., Lubiw, A. (eds.) *GD 2015*. LNCS, vol. 9411, pp. 16–29. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27261-0_2
15. Cruz, I.F., Twarog, J.P.: 3D graph drawing with simulated annealing. In: Brandenburg, F.J. (ed.) *GD 1995*. LNCS, vol. 1027, pp. 162–165. Springer, Heidelberg (1996). <https://doi.org/10.1007/BFb0021800>
16. Davidson, R., Harel, D.: Drawing graphs nicely using simulated annealing. *ACM Trans. Graphics* **15**(4), 301–331 (1996)
17. Diehl, S., Görg, C.: Graphs, they are changing. In: Goodrich, M.T., Kobourov, S.G. (eds.) *GD 2002*. LNCS, vol. 2528, pp. 23–31. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36151-0_3
18. Diehl, S., Görg, C., Kerren, A.: Preserving the mental map using foresighted layout. In *Eurographics/IEEE VGTC Symposium on Visualization (VisSym 2001)*, pp. 175–184 (2001)
19. Dwyer, T., Eades, P.: Visualising a fund manager flow graph with columns and worms. In: *Proceedings of the International Conference on Information Visualisation (IV 2002)*, pp. 147–152 (2002)

20. Erten, C., Harding, P.J., Kobourov, S.G., Wampler, K., Yee, G.: GraphAEL: graph animations with evolving layouts. In: Liotta, G. (ed.) GD 2003. LNCS, vol. 2912, pp. 98–110. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24595-7_9
21. Erten, C., Kobourov, S.G., Le, V., Navabi, A.: Simultaneous graph drawing: layout algorithms and visualization schemes. In: Liotta, G. (ed.) GD 2003. LNCS, vol. 2912, pp. 437–449. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24595-7_41
22. Farrugia, M., Quigley, A.: Effective temporal graph layout: a comparative study of animation versus static display methods. *J. Inform. Vis.* **10**(1), 47–64 (2011)
23. Forrester, D., Kobourov, S.G., Navabi, A., Wampler, K., Yee, G.V.: Graphael: a system for generalized force-directed layouts. In: Pach, J. (ed.) GD 2004. LNCS, vol. 3383, pp. 454–464. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31843-9_47
24. Frick, A., Ludwig, A., Mehldau, H.: A fast adaptive layout algorithm for undirected graphs (extended abstract and system demonstration). In: Tamassia, R., Tollis, I.G. (eds.) GD 1994. LNCS, vol. 894, pp. 388–403. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-58950-3_393
25. Frishman, Y., Tal, A.: Online dynamic graph drawing. *IEEE Trans. Vis. Comput. Graphics* **14**(4), 727–740 (2008)
26. Gajer, P., Kobourov, S.G.: GRIP: graph drawing with intelligent placement. In: Marks, J. (ed.) GD 2000. LNCS, vol. 1984, pp. 222–228. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44541-2_21
27. Gansner, E.R., Hu, Y.: A maxent-stress model for graph layout. *IEEE Trans. Vis. Comput. Graphics* **19**(6), 927–940 (2013)
28. Gorochofski, T.E., Di Bernardo, M., Grierson, C.S.: Using aging to visually uncover evolutionary processes on networks. *IEEE Trans. Vis. Comput. Graphics* **18**(8), 1343–1352 (2012)
29. Grayson, S., Wade, K., Meaney, G., Greene, D.: Sensibility of different sliding windows in constructing co-occurrence networks from literature. In: Bozic, B., Mendel-Gleason, G., Debryne, C., O’Sullivan, D. (eds.) Proceedings of 2nd International Workshop on Computational History and Data-Driven Humanities, pp. 65–77 (2016)
30. Groh, G., Hanstein, H., Wörndl, W.: Interactively visualizing dynamic social networks with DySoN. In: Workshop on Visual Interfaces to the Social and the Semantic Web (VISSW 2009), February 2009
31. Hachul, S., Jünger, M.: Drawing large graphs with a potential-field-based multilevel algorithm. In: Pach, J. (ed.) GD 2004. LNCS, vol. 3383, pp. 285–295. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31843-9_29
32. Itoh, M., Yoshinaga, N., Toyoda, M., Kitsuregawa, M.: Analysis and visualization of temporal changes in bloggers’ activities and interests. In: IEEE Pacific Visualization Symposium (PacificVis 2012), pp. 57–64 (2012)
33. Kobourov, S.G., Pupyrev, S., Saket, B.: Are crossings important for drawing large graphs? In: Duncan, C., Symvonis, A. (eds.) GD 2014. LNCS, vol. 8871, pp. 234–245. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45803-7_20
34. Landgraf, B.: 3D graph drawing. In: Kaufmann, M., Wagner, D. (eds.) Drawing Graphs. LNCS, vol. 2025, pp. 172–192. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44969-8_7
35. Liu, G., Austen, E.L., Booth, K.S., Fisher, B.D., Argue, R., Rempel, M.I., Enns, J.T.: Multiple-object tracking is based on scene, not retinal, coordinates. *J. Exp. Psychol. Hum. Percept. Perform.* **31**(2), 235–247 (2005)

36. Liu, Q., Hu, Y., Shi, L., Mu, X., Zhang, Y., Tang, J.: EgoNetCloud: event-based egocentric dynamic network visualization. In: Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST 2015), pp. 65–72. IEEE Computer Society (2015)
37. Misue, K., Eades, P., Lai, W., Sugiyama, K.: Layout adjustment and the mental map. *J. Visual Lang. Comput.* **6**(2), 183–210 (1995)
38. Munzner, T.: H3: laying out large directed graphs in 3D hyperbolic space. In: Proceedings of the IEEE Symposium on Information Visualization (InfoVis 1997), pp. 2–10, 1997
39. Newcomb, T.M.: *The Acquaintance Process*. New York, Holt, Reinhard & Winston (1961)
40. Ortman, M., Klimenta, M., Brandes, U.: A sparse stress model. In: Hu, Y., Nöllenburg, M. (eds.) GD 2016. LNCS, vol. 9801, pp. 18–32. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_2
41. Parker, G., Franck, G., Ware, C.: Visualization of large nested graphs in 3D: navigation and interaction. *J. Visual Lang. Comput.* **9**(3), 299–317 (1998)
42. Pylyshyn, Z.W., Storm, R.W.: Tracking multiple independent targets: Evidence for a parallel tracking mechanism. *Spat. Vis.* **3**(3), 179–197 (1988)
43. Rauber, P.E., Falcão, A.X., Telea, A.C.: Visualizing time-dependent data using dynamic t-SNE. In: Bertini, E., Elmqvist, N., Wischgoll, T. (eds.) EuroVis 2016, Short Papers. Eurographics Association (2016)
44. Sallaberry, A., Muelder, C., Ma, K.-L.: Clustering, visualizing, and navigating for large dynamic graphs. In: Didimo, W., Patrignani, M. (eds.) GD 2012. LNCS, vol. 7704, pp. 487–498. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36763-2_43
45. Shneiderman, B., Aris, A.: Network visualization by semantic substrates. *IEEE Trans. Vis. Comput. Graphics (InfoVis 2006)* **12**(5), 733–740 (2006)
46. Simonetto, P., Archambault, D., Auber, D., Bourqui, R.: ImPrEd: an improved force-directed algorithm that prevents nodes from crossing edges. *Comput. Graphics Forum (EuroVis 2011)* **30**(3), 1071–1080 (2011)
47. Simonetto, P., Archambault, D., Scheidegger, C.: A simple approach for boundary improvement of Euler diagrams. *IEEE Trans. Vis. Comput. Graphics (InfoVis 2015)* **22**(1), 678–687 (2016)
48. Tversky, B., Morrison, J., Betrancourt, M.: Animation: can it facilitate? *Int. J. Hum Comput Stud.* **57**(4), 247–262 (2002)
49. van den Elzen, S., Holten, D., Blaas, J., van Wijk, J.J.: Dynamic network visualization with extended massive sequence views. *IEEE Trans. Vis. Comput. Graphics* **20**(8), 1087–1099 (2014)
50. van den Elzen, S., Holten, D., Blaas, J., van Wijk, J.J.: Reducing snapshots to points: a visual analytics approach to dynamic network exploration. *IEEE Trans. Vis. Comput. Graphics* **22**(1), 1–10 (2016)

Point-Set Embeddings

Colored Point-Set Embeddings of Acyclic Graphs

Emilio Di Giacomo^{1(✉)}, Leszek Gasieniec², Giuseppe Liotta¹,
and Alfredo Navarra¹

¹ Università degli Studi di Perugia, Perugia, Italy
{emilio.digiacomo,giuseppe.liotta,alfredo.navarra}@unipg.it

² University of Liverpool, Liverpool, UK
L.A.Gasieniec@liverpool.ac.uk

Abstract. We show that any planar drawing of a forest of three stars whose vertices are constrained to be at fixed vertex locations may require $\Omega(n^{\frac{2}{3}})$ edges each having $\Omega(n^{\frac{1}{3}})$ bends in the worst case. The lower bound holds even when the function that maps vertices to points is not a bijection but it is defined by a 3-coloring. In contrast, a constant number of bends per edge can be obtained for 3-colored paths and for 3-colored caterpillars whose leaves all have the same color. Such results answer to a long standing open problem.

1 Introduction

A pioneering paper by Pach and Wenger [9] studied the problem of computing a planar drawing of a graph G with the constraint that the mapping of the vertices to the points in the plane, that represent the vertices of G , is given as part of the input. Pach and Wenger proved that, for any given mapping, a planar graph with n vertices admits a planar drawing such that the curve complexity, i.e. the number of bends per edge, is $O(n)$. Furthermore, they proved that the bound on the curve complexity is (almost surely) tight as n tends to infinity when G has $O(n)$ independent edges. This implies that the curve complexity of a planar drawing with vertices at fixed locations may be $\Omega(n)$ even for structurally very simple graphs such as paths or matchings, for which the number of independent edges is linear in n .

These results have motivated the study of a relaxed version of the problem where the function that associates vertices of the graph to points of the plane is not a bijection. Namely, an instance of the k -colored point set embeddability problem receives as input an n -vertex planar graph G such that every vertex is

The work has been supported in part by the European project “Geospatial based Environment for Optimisation Systems Addressing Fire Emergencies” (GEO-SAFE), contract no. H2020-691161, by the Network Sciences and Technologies (NeST) initiative at University of Liverpool, and by the Italian project: “RISE: un nuovo framework distribuito per data collection, monitoraggio e comunicazioni in contesti di emergency response”, Fondazione Cassa Risparmio Perugia, code 2016.0104.021.

given one of k distinct colors and a set S of n distinct points, such that, each point is given one of the k distinct colors. The number of points of S having a certain color i is the same as the number of vertices of G having color i . The goal is to compute a planar drawing of G with curve complexity independent of n where every vertex of a specific color is represented by a point of the same color. When $k = n$ the k -colored point set embeddability problem coincides with the problem of computing a drawing with vertices at fixed locations and thus the lower bounds by Pach and Wenger hold. Therefore several papers have focused on small values of k (typically $k \leq 3$) to see whether better bounds on the curve complexity could be achieved in this scenario (see, e.g., [1, 4–7]).

For $k = 1$, Kaufmann and Wiese [8] proved that every planar graph admits a 1-colored point set embedding onto any point set with curve complexity at most 2. For $k = 2$, outerplanar graphs always admit a 2-colored point set embedding with $O(1)$ curve complexity [2]. However, for $k \geq 2$, there are 2-connected k -colored planar graphs for which a k -colored point set embedding may require $\Omega(n)$ bends on $\Omega(n)$ edges [1]. This result extends the lower bound of Pach and Wenger [9] to a much more relaxed set of constraints on the location of the vertices, but it does so by using 2-connected graphs instead of just (not necessarily connected) planar graphs. For example, the problem of establishing tight bounds on the curve complexity of k -colored forests for small values of $k \geq 3$ is a long standing open problem (see, e.g., [1]). We explicitly address this gap in the literature and consider the k -colored point set embeddability problem for acyclic graphs and $k \geq 3$. Our main results are as follows.

- In Sect. 3, we prove that a planar drawing of a forest of three stars and n vertices may require $\Omega(n^{\frac{2}{3}})$ edges with $\Omega(n^{\frac{1}{3}})$ bends each, even if the mapping of the vertices to the points is defined by using a set of k colors with $k \geq 3$. In contrast, a constant number of bends per edge can always be achieved if the number of stars is at most two (for any number of colors) or the number of colors is at most two (for any number of stars).
- Since the above result implies that 3-colored point set embeddings of 3-colored caterpillars may have a non-constant curve complexity, in Sect. 4 we study subfamilies of 3-colored caterpillars for which constant curve complexity is possible. We prove that every 3-colored path and every 3-colored caterpillar whose leaves all have the same color admit a 3-colored point-set embedding with constant curve complexity onto any 3-colored point set.
- Finally, still in Sect. 4, we prove that any 4-colored path π such that the vertices of colors 0 and 1 precede all vertices of colors 2 and 3 when moving along π has a 4-colored point set embedding with at most five bends per edge onto any 4-colored point-set.

Concerning the lower bound, it is worth mentioning that the argument by Pach and Wenger [9] does not apply to families of graphs where the number of independent edges is not a function of n . Hence, our lower bound extends the one by Pach and Wenger about the curve complexity of planar drawings with vertices at fixed locations also to those graphs for which the number of

independent edges does not grow with n . For reasons of space some proofs are omitted in this paper and can be found in [3].

2 Preliminaries

Let $G = (V, E)$ be a graph. A k -coloring of G is a partition $\{V_0, V_1, \dots, V_{k-1}\}$ of V . The integers $0, 1, \dots, k-1$ are called *colors* and G is called a k -colored graph. For each vertex $v \in V_i$ we denote by $col(v)$ the color i of v .

Let S be a set of distinct points in the plane. For any point $p \in S$, we denote by $x(p)$ and $y(p)$ the x - and y -coordinates of p , respectively. We denote by $CH(S)$ the convex hull of S . Throughout the paper we always assume that the points of S have different x -coordinates (if not we can rotate the plane so to achieve this condition). A k -coloring of S is a partition $\{S_0, S_1, \dots, S_{k-1}\}$ of S . A set of points S with a k -coloring is called a k -colored point set. For each point $p \in S_i$, $col(p)$ denotes the color i of p . A k -colored point set S is *compatible with a k -colored graph G* if $|V_i| = |S_i|$ for every i . If G is planar we say that G has a *topological point-set embedding* on S if there exists a planar drawing of G such that: (i) every vertex v is mapped to a distinct point p of S with $col(p) = col(v)$, (ii) each edge e of G is drawn as simple Jordan arc. We say that G has a *k -colored point-set embedding* on S if there exists a planar drawing of G such that: (i) every vertex v is mapped to a distinct point p of S with $col(p) = col(v)$, (ii) each edge e of G is drawn as a polyline λ . A point shared by any two consecutive segments of λ is called a *bend* of e . The maximum number of bends along an edge is the *curve complexity* of the k -colored point-set embedding. A k -colored sequence σ is a sequence of (possibly repeated) colors c_0, c_1, \dots, c_{n-1} such that $0 \leq c_j \leq k-1$ ($0 \leq j \leq n-1$). We say that σ is *compatible with a k -colored graph G* if color i occurs $|V_i|$ times in σ . Let S be a k -colored point set. Let p_0, \dots, p_{n-1} be the points of S with $x(p_0) < \dots < x(p_{n-1})$. The k -colored sequence $col(p_0), \dots, col(p_{n-1})$ is called the *k -colored sequence induced by S* , and is denoted as $seq(S)$. A set of points S is *one-sided convex* if they are in convex position and the two points with minimum and maximum x -coordinate are consecutive along $CH(S)$. In a k -colored one-sided convex point set, the sequence of colors encountered clockwise along $CH(S)$, starting from the point with minimum x -coordinate, coincides with $seq(S)$.

A *Hamiltonian cycle* of a graph G is a simple cycle that contains all vertices of G . A graph G that admits a Hamiltonian cycle is said to be *Hamiltonian*. A planar graph G is *sub-Hamiltonian* if either G is Hamiltonian or G can be augmented with dummy edges (but not with dummy vertices) to a graph $aug(G)$ that is Hamiltonian and planar. A *subdivision* of a graph G is a graph obtained from G by replacing each edge by a path with at least one edge. Internal vertices on such a path are called *division vertices*. Every planar graph has a subdivision that is sub-Hamiltonian. Let G be a planar graph and let $sub(G)$ be a sub-Hamiltonian subdivision of G . The graph $aug(sub(G))$ is called a *Hamiltonian augmentation of G* and will be denoted as $Ham(G)$. Let \mathcal{C} be the Hamiltonian cycle of a Hamiltonian augmentation $Ham(G)$ of G . Let e be an edge of \mathcal{C} ,

let $\mathcal{P} = \mathcal{C} \setminus e$ be the Hamiltonian path obtained by removing e from \mathcal{C} , and let $v_0, v_1, \dots, v_{n'-1}$ be the vertices of G in the order they appear along \mathcal{P} . Finally, let $\sigma = c_0, c_1, \dots, c_{n'-1}$ be a k -colored sequence. \mathcal{P} is a k -colored Hamiltonian path consistent with σ if $\text{col}(v_i) = c_i$ ($0 \leq i \leq n' - 1$). \mathcal{C} is a k -colored Hamiltonian cycle consistent with σ if there exists an edge $e \in \mathcal{C}$ such that $\mathcal{P} = \mathcal{C} \setminus e$ is a k -colored Hamiltonian path consistent with σ . $\text{Ham}(G)$ is called a k -colored Hamiltonian augmentation of G consistent with σ . The following theorem has been proved in [2] (see also [1,6]).

Theorem 1. [2] *Let G be a k -colored planar graph and S be a k -colored point set consistent with G . If G has a k -colored Hamiltonian augmentation consistent with $\text{seq}(S)$ and at most d division vertices per edge then G admits a k -colored point-set embedding on S with at most $2d + 1$ bends per edge.*

The next lemma can be easily derived from Theorem 1.

Lemma 1. *Let G be a k -colored graph, and S be a k -colored one-sided convex point set compatible with G . If G has a topological k -colored point-set embedding on S such that each edge crosses $CH(S)$ at most b times, then G admits a k -colored point-set embedding on S with at most $2b + 1$ bends per edge.*

Let $G = (V, E)$ be a planar graph. A *topological book embedding* of G is a planar drawing such that all vertices of G are represented as points of a horizontal line ℓ , called the *spine*. Each of the half-planes defined by ℓ is a *page*. Each edge of a topological book embedding is either in the top page, or completely in the bottom page, or it can be on both pages, in which case it crosses the spine. Each crossing between an edge and the spine is called a *spine crossing*. It is also assumed that in a topological book embedding every edge consists of one or more circular arcs, such that no two consecutive arcs are in the same page¹. Let G be a k -colored graph and let σ be a k -colored sequence compatible with G . A topological book embedding of G is consistent with σ if the sequence of vertex colors along the spine coincides with σ . Let S be a k -colored point set compatible with a k -colored planar graph G and let $\text{seq}(S)$ be the k -colored sequence induced by S . The following lemma can be proved similarly to Lemma 1.

Lemma 2. *If G admits a topological book embedding consistent with $\text{seq}(S)$ and having at most h spine crossing per edge, then G admits a point-set embedding on S with curve complexity at most $2h + 1$.*

3 Point-Set Embeddings of Stars

In this section we establish that a 3-colored point-set embedding of a forest of three stars may require $\Omega(n^{\frac{1}{3}})$ bends along $\Omega(n^{\frac{2}{3}})$ edges by exploiting a previous

¹ The more general concept of *h-page topological book embedding* exists, where each arc can be drawn on one among h different pages. For simplicity we use the term topological book embedding to mean 2-page topological book embedding.

result about biconnected outerplanar graphs. We start by recalling the result in [2]. An *alternating point set* S_n is a 3-colored one-sided convex point set such that: (i) S_n has n points for each color 0, 1, and 2, and (ii) when going along the convex hull $CH(S_n)$ of S_n in clockwise order, the sequence of colors encountered is 0, 1, 2, 0, 1, 2, . . . Each set of consecutive points colored 0, 1, 2 is called a *triplet*.

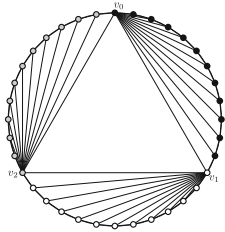


Fig. 1. A 3-fan G_n for $n = 12$.

A *3-fan*, denoted as G_n , is a 3-colored outerplanar graph with $3n$ vertices ($n \geq 2$) and defined as follows. G_n consists of a simple cycle formed by n vertices of color 0, followed (in the counterclockwise order) by n vertices of color 1, followed by n vertices of color 2. The vertex of color i adjacent in the cycle to a vertex of color $i - 1$ (indices taken modulo 3) is denoted as v_i . Also, in G_n every vertex colored i is adjacent to v_i ($i = 0, 1, 2$) and vertices v_0, v_1, v_2 form a 3-cycle of G_n . See, e.g. Fig. 1. The following theorem has been proved in [2].

Theorem 2. [2] *Let h be a positive integer and let G_n be a 3-fan for $n \geq 79h^3$, and let S_n be an alternating point set compatible with G_n . In every 3-colored point-set embedding of G_n on S_n there is one edge with more than h bends.*

The forest of stars that we use to establish our lower bound is called a *3-sky* and is denoted by F_n . It consists of three stars T_0, T_1, T_2 such that: (i) each T_i ($i = 0, 1, 2$) has n vertices ($n \geq 2$); (ii) all the vertices of each T_i ($i = 0, 1, 2$) have the same color i .

Let Γ_n be a point-set embedding of F_n on S_n . An *uncrossed triplet* of Γ_n is a triplet p_i, p_{i+1}, p_{i+2} of points of S_n such that, when moving along $CH(S_n)$ in clockwise order, no edge of Γ_n crosses $CH(S_n)$ between p_i and p_{i+1} and between p_{i+1} and p_{i+2} . A triplet is *crossed k times* if the total number of times that $CH(S_n)$ is crossed by some edges between p_i and p_{i+1} and between p_{i+1} and p_{i+2} is k . A *leaf triplet* of Γ_n is a triplet of S_n whose points represent leaves of F_n . Analogously, a *root triplet* is a triplet of S_n whose points represent the three roots of F_n . The following lemma establishes the first relationship between the curve complexity of some special types of 3-colored point-set embeddings of F_n and those of a 3-fan G_n .

Lemma 3. *Let F_n be a 3-sky, S_n be an alternating point set compatible with F_n , and Γ_n be a 3-colored topological point-set embedding of F_n on S_n . If Γ_n has an uncrossed leaf triplet and each edge of Γ_n crosses $CH(S_n)$ at most b times, then the 3-fan G_n has a 3-colored topological point-set embedding on S_n such that each edge crosses $CH(S_n)$ at most $3b + 2$ times.*

Proof. We show how to use Γ_n to construct a topological point-set embedding of the 3-fan G_n on S_n with at most $3b + 2$ crossings of $CH(S_n)$ per edge.

Let p_j, p_{j+1}, p_{j+2} be an uncrossed leaf triplet. Every point of the triplet represents a leaf of a different star (because they have different color). Denote by q_i the point of Γ_n representing the root of T_i ($i = 0, 1, 2$) and denote by e_i the edge connecting q_i to p_{j+i} . The idea is to connect the three points q_0, q_1, q_2

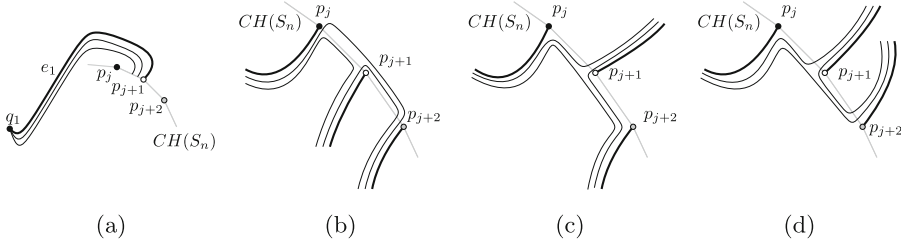


Fig. 2. Insertion of a cycle connecting q_0 , q_1 and q_2 . (a) Drawing of the curves following the edge e_1 . (b), (c), and (d) Connection of the six curves to form a cycle.

with a 3-cycle that does not cross any existing edges. For each edge e_i we draw two curves that from q_i run very close to e_i until they reach $CH(S_n)$. The two curves are drawn on the same side of e_i such that they are consecutive in the circular order of the edges around q_i (see Fig. 2(a) for an illustration). These two curves do not intersect any existing edges and cross $CH(S_n)$ the same number of times as e_i . The six drawn curves are now suitably connected to realize a cycle C connecting q_0, q_1, q_2 . Depending on which side the various curves reach $CH(S_n)$, the connections are different. However in all cases we can connect two curves to form a single edge by crossing $CH(S_n)$ at most two additional times and without violating planarity (see Fig. 2(b), (c) and (d)). Thus, we have added to Γ_n three edges e'_i connecting q_i to q_{i+1} (indices taken modulo 3), each crossing $CH(S_n)$ at most $2b + 2$ times. Also, since the two curves that follow an edge e_i are both drawn on the same side of e_i , the cycle C does not have any vertices inside. Notice that, depending on the case with respect to the connection of the curves, q_0, q_1 , and q_2 appear along C either in the clockwise or in the counterclockwise order. W.l.o.g. we assume that the clockwise order is q_0, q_1 , and q_2 .

The obtained drawing is not yet a topological point-set embedding of G_n because the cycle C' connecting all the vertices is missing. We first add edges connecting leaves of the same color. Let $e'_i = \bar{e}_0, \bar{e}_1, \dots, \bar{e}_{n-2} = e'_{i-1}$ be the edges incident to q_i in the circular order around q_i (this is the counterclockwise order under our assumption that q_0, q_1 , and q_2 are located in the clockwise order along C). We add an edge between the leaf of \bar{e}_j and the leaf of \bar{e}_{j+1} (for $j = 0, 1, \dots, n - 3$) as follows. Starting from the leaf of \bar{e}_j , we draw a curve following the edge \bar{e}_j until we arrive very close to q_i and then we follow \bar{e}_{j+1} until we reach the leaf of \bar{e}_{j+1} . The added edges do not cross any existing edges and cross $CH(S_n)$ a number of times equal to the number of times that \bar{e}_j crosses $CH(S_n)$ plus the number of times that \bar{e}_{j+1} crosses $CH(S_n)$, so at most $2b$.

It remains to add the edges of C' connecting vertices of different colors. There are three such edges and they connect vertex v_i ($i = 0, 1, 2$) of G_n to a vertex of color $i + 1$ (indices taken modulo 3). We add an edge connecting q_i to a leaf of color $i + 1$ as follows. Let e''_{i+1} be the edge incident to q_{i+1} that follows e'_i in the clockwise order around q_{i+1} (this is an edge connecting q_{i+1} to a leaf of color $i + 1$). Starting from q_i we draw a curve following the edge e'_i until we

arrive very close to q_{i+1} and then we follow e''_{i+1} until we reach the leaf of e''_{i+1} . The constructed curve connects q_i to a leaf of color $i + 1$ and does not cross any existing edge. It crosses $CH(S_n)$ at most the number of times that e'_{i+1} crosses $CH(S_n)$ (that is $2b + 2$) plus the number of times that e''_{i+1} crosses $CH(S_n)$ (that is b). Thus the total number of crossing of $CH(S_n)$ is at most $3b + 2$. \square

The next two lemmas explain how to obtain a 3-colored topological book embedding that satisfies Lemma 3.

Lemma 4. *Let F_n be a 3-sky, S_n be an alternating point set compatible with F_n , and Γ_n be a 3-colored topological point-set embedding of F_n on S_n with a root triplet. If Γ_n has a leaf triplet τ that is crossed c times ($c < n$) and each edge crosses $CH(S_n)$ at most b times, then there exists a 3-sky $F_{n'}$ which is a subgraph of F_n and an alternating point set $S_{n'}$ which is a subset of S_n such that: (i) $n' \geq n - c$; (ii) there exists a 3-colored topological point-set embedding $\Gamma_{n'}$ of $F_{n'}$ on $S_{n'}$ such that each edge crosses $CH(S_{n'})$ at most $b + 1$ times; (iii) τ is an uncrossed leaf triplet of $\Gamma_{n'}$.*

Lemma 5. *Let F_n be a 3-sky, S_n be an alternating point set compatible with F_n , and Γ_n be a 3-colored topological point-set embedding of F_n on S_n . If each edge of Γ_n crosses $CH(S_n)$ at most b times, then there exists a 3-sky $F_{n'}$ which is a subgraph of F_n and an alternating point set $S_{n'}$ which is a subset of S_n such that: (i) $n' \geq \frac{n}{3} - 3$; (ii) there exists a 3-colored topological point-set embedding $\Gamma_{n'}$ of $F_{n'}$ on $S_{n'}$ such that each edge crosses $CH(S_{n'})$ at most $b + 2$ times; (iii) $\Gamma_{n'}$ has a root triplet.*

Lemma 6. *Let h be a positive integer, F_n be a 3-sky for $n = 520710h^3$, and S_n be an alternating point set compatible with F_n . In every 3-colored point-set embedding of F_n on S_n there exist at least h^2 edges with more than h bends.*

Proof (sketch). Let $F_{n_i}, i = 1, 2, \dots, h^2$, be a 3-sky for $n_i = 520689h^3 + 21h \cdot i$ and let S_{n_i} be an alternating point set compatible with F_{n_i} . We prove by induction on i that in every 3-colored point-set embedding of F_{n_i} on S_{n_i} there exist i edges with more than h bends. Notice that for $i = h^2$, we have $n_i = n$.

Base case: $i = 1$: We have to prove that in any 3-colored point-set embedding of F_{n_1} on S_{n_1} with $n_1 = 520689h^3 + 21h$, there exists one edge with more than h bends. Suppose as a contradiction that there exists a 3-colored point-set embedding Γ_{n_1} of F_{n_1} on S_{n_1} with curve complexity h . Γ_{n_1} is also a 3-colored topological point-set embedding of F_{n_1} on S_{n_1} such that each edge crosses $CH(S_{n_1})$ at most $2h$ times (each edge consists of at most $h + 1$ segments). By Lemma 5 there exists a 3-colored point-set embedding $\Gamma_{n'}$ of a 3-sky $F_{n'}$ on an alternating point set $S_{n'}$ such that: (i) $n' \geq \frac{n_1}{3}$; (ii) each edge of $\Gamma_{n'}$ crosses $CH(S_{n'})$ at most $2h + 2$ times; (iii) $\Gamma_{n'}$ has a root triplet.

Since each edge of $\Gamma_{n'}$ crosses $CH(S_{n'})$ at most $2h + 2$ times and there are $3(n' - 1) \geq n_1 - 3$ edges in total, there are at most $(2h + 2)(n_1 - 3)$ crossings of $CH(S_{n_1})$ in total. The number of leaf triplets in $\Gamma_{n'}$ is $n' - 1 \geq \frac{n_1}{3} - 1$. It follows that there is at least one leaf triplet τ crossed at most $\frac{3(2h+2)(n_1-3)}{(n_1-3)} = 6h+6 \leq 7h$

times. By Lemma 4 there exists a 3-colored point-set embedding $\Gamma_{n''}$ of a 3-sky $F_{n''}$ on an alternating point set $S_{n''}$ such that: (i) $n'' \geq n' - 7h$; (ii) each edge of $\Gamma_{n''}$ crosses $S_{n''}$ at most $2h + 3$ times; (iii) τ is uncrossed. By Lemma 3, the 3-fan $G_{n''}$ has a 3-colored topological point-set embedding on $S_{n''}$ such that each edge crosses $CH(S_{n''})$ at most $6h + 11$ times and by Lemma 1 a 3-colored point set embedding with curve complexity at most $12h + 23$. On the other hand, since $n_1 = 520689h^3 + 21h$, we have that $n'' \geq n' - 7h \geq \frac{n_1}{3} - 7h = \frac{520689h^3 + 21h}{3} - 7h = \frac{520689}{3}h^3 \geq \frac{520689}{3}h^3 = 79(13h)^3$ and by Theorem 2, in every 3-colored point-set embedding of $G_{n''}$ on $S_{n''}$ at least one edge that has more than $13h$ bends – a contradiction.

Inductive step: $i > 1$. We have to prove that in any 3-colored point-set embedding of F_{n_i} on S_{n_i} with $n_i = 520689h^3 + 21h \cdot i$, there exist i edges with more than h bends.

We first prove that there exists at least one edge with more than h bends. Suppose as a contradiction that there exists a 3-colored point-set embedding Γ_{n_i} of F_{n_i} on S_{n_i} with curve complexity h . With the same reasoning as in the base case, there would exist a 3-colored point set embedding with curve complexity at most $12h + 23$ of a 3-fan $G_{n''}$, with $n'' \geq \frac{n_i}{3} - 7h$. Since $n_i = 520689h^3 + 21h \cdot i$, we have that $n'' \geq \frac{n_i}{3} - 7h = \frac{520689h^3 + 21h \cdot i}{3} - 7h = \frac{520689}{3}h^3 + 7h(i - 1) \geq \frac{520689}{3}h^3 = 79(13h)^3$ and by Theorem 2, in every 3-colored point-set embedding of $G_{n''}$ on $S_{n''}$ at least one edge has more than $13h$ bends – again a contradiction.

This proves that there is at least one edge e crossed more than h times. We now remove this edge and the whole triplet that contains the point representing the leaf of e . We then arbitrarily remove $21h - 1$ triplets. The resulting drawing is a 3-colored point-set embedding $\Gamma_{n'''}$ of $F_{n'''}$ on $S_{n'''}$ for $n''' = n_{i-1}$. By induction, it contains $i - 1$ edges each having more than h bends. It follows that Γ_{n_i} has i edges each having more than h bends. Since for $i = h^2$ we have $n_i = n$, the statement follows. \square

Theorem 3. *For sufficiently large n , there exists a 3-colored forest F_n consisting of three monochromatic stars with n vertices and a 3-colored point set S_n in convex position compatible with F_n such that any 3-colored point-set embedding of F_n on S_n has $\Omega(n^{\frac{2}{3}})$ edges having $\Omega(n^{\frac{1}{3}})$ bends.*

We conclude this section with some results deriving from Theorem 3 and/or related to it. Firstly, Theorem 3 extends the result of Theorem 2 since it implies that a 3-colored point set embedding of G_n may require $\Omega(n^{\frac{2}{3}})$ edges with $\Omega(n^{\frac{1}{3}})$ bends each. Moreover, the result of Theorem 3 implies an analogous result for a k -colored forest of at least three stars for every $k \geq 3$. In particular, when $k = n$ we have the following result that extends the one by Pach and Wenger [9].

Corollary 1. *Let F be a forest of three n -vertex stars. Every planar drawing of F with vertices at fixed vertex locations has $\Omega(n^{\frac{2}{3}})$ edges with $\Omega(n^{\frac{1}{3}})$ bends each.*

One may wonder whether the lower bound of Theorem 3 also holds when the number of colors or the number of stars is less than three. However, it is immediate to see that this is not the case, i.e., the following theorem holds.

Theorem 4. *Let F be a k -colored forest of h stars and S be a set of points compatible with F . If $\max\{k, h\} = 2$ then F has a k -colored point-set embedding on S with curve complexity at most 2.*

Since a caterpillar can be regarded as a set of stars whose roots are connected in a path, the lower bound of Theorem 3 also holds for caterpillars. This answers an open problem in [1] about the curve complexity of k -colored point-set embeddings of trees for $k \geq 3$. Note that $O(1)$ curve complexity for 2-colored outerplanar graphs has been proved in [2].

Corollary 2. *For sufficiently large n , a 3-colored point-set embedding of a 3-colored caterpillar may require $\Omega(n^{\frac{2}{3}})$ edges having $\Omega(n^{\frac{1}{3}})$ bends.*

4 Point-Set Embeddings of Paths and Caterpillars

In the light of Corollary 2, one may ask whether there exist subclasses of 3-colored caterpillars for which constant curve complexity can be guaranteed. In this section we first prove that this is the case for 3-colored paths and then we extend the result to 3-colored caterpillars whose leaves all have the same color.

Based on Lemma 2, we prove that a 3-colored path P has a topological book embedding consistent with $seq(S)$ and having a constant number of spine crossings. Namely, we first remove the vertices and points of one color from P and S , obtaining a 2-colored path P' and a compatible 2-colored point set S' . Next, we construct a topological book embedding $\gamma_{P'}$ of P' consistent with $seq(S')$ with at most two spine crossings per edge and with suitable properties. Then we use such properties to reinsert the third color and obtain a topological book embedding of P consistent with $seq(S)$.

P' and $\sigma' = seq(S')$ can be regarded as two binary strings of the same size where one color is represented by bit 0 and the other one by bit 1. P' and σ' are *balanced* if the number of 0's (1's, resp.) in P' equals the number of 0's (1's, resp.) in σ' . P' and σ' are a *minimally balanced pair* if there does not exist a prefix of P' and a corresponding prefix of σ' that are balanced.

Lemma 7. *Let P and σ be a minimally balanced pair of length $k > 1$. Let $b_j(P)$ denote the j -th bit of P and $b_j(\sigma)$ denote the j -th bit of σ . Then $b_1(P) \neq b_k(P)$, $b_k(P) = b_1(\sigma)$, and $b_1(P) = b_k(\sigma)$.*

Let Γ be a topological book embedding, ℓ be the spine of Γ , and p be a point of ℓ (possibly representing a vertex). We say that p is visible from above (below) if the vertical ray with origin at p and lying in the top (bottom) page does not intersect any edge of Γ . We say that the segment \overline{pq} is *visible from above (below)* if each point r in the segment is visible from above (below). Let u and v be two vertices of Γ that are consecutive along the spine ℓ , we say that segment \overline{uv} is *accessible* if it contains a segment that is visible from below. A vertex v of Γ is *hook visible* if there exists a segment \overline{pq} of the spine such that \overline{pq} is visible from below and for any point r of \overline{pq} we can add an edge in the top page of Γ

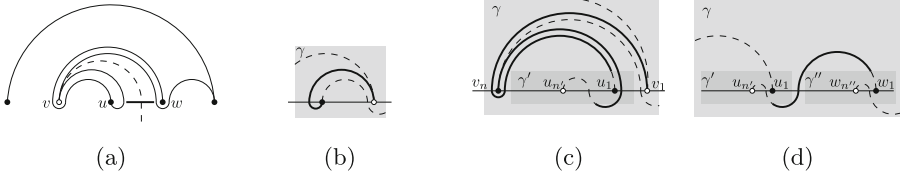


Fig. 3. (a) Illustration of the hook visibility property. The bold segment is the access interval. (b)-(d) Proof of Lemma 8: (b) Base cas; (c) Case 1; (d) Case 2.

connecting v with r without crossing any other edges of Γ (see Fig. 3(a)); \overline{pq} is the *access interval for vertex v* . If the access interval is to the right (left) of w we say that v is *hook visible from the right (left)*.

Lemma 8. *Let P be a 2-colored path and σ be a 2-colored sequence compatible with P . Path P admits a topological book embedding γ consistent with σ and with the following properties: (a) Every edge of γ crosses the spine at least once and at most twice. (b) For any two vertices u and v that are consecutive along the spine ℓ of γ , segment \overline{uv} is accessible from below. (c) Every spine crossing is visible from below. (d) The first vertex v_1 of P is visible from above; the last vertex v_n of P is hook visible from the right; to the right of its access interval there is only one vertex and no spine crossing.*

Proof. We prove the statement by induction on the length n of P (and of σ). If $n = 1$ the statement trivially holds. If $n = 2$ we draw the unique edge of P with one spine crossing immediately to the left of the leftmost vertex in the drawing (see Fig. 3(b)). Also in this case the statement holds. Suppose that $n > 2$ and that the statement holds for every $k < n$. We distinguish between two cases.

Case 1: P and σ are a minimally balanced pair. By Lemma 7 the first vertex of P has the same color as the last element of σ , the last element of P has the same color as the first element of σ and these two colors are different. This means that by removing the first and the last elements from both P and σ , we obtain a new 2-colored path P' of length $n - 2$ and a new 2-colored sequence σ' compatible with P' . By induction, P' admits a topological book embedding γ' consistent with σ' and satisfying properties (a)–(d). To create a topological book embedding of P consistent with σ , we add a point p_1 before all the points of γ' , whose color is the same as the last vertex v_n of P , and a point p_2 after all points of γ' , whose color is the same as the first vertex v_1 of P . Vertex v_1 is mapped to p_2 and vertex v_n is mapped to p_1 . We connect v_1 to the first vertex u_1 of P' with an edge incident to p_2 from above, crossing the spine once immediately before p_1 and once immediately after p_1 and arriving to u_1 from above (by property (d), u_1 is visible from above). We then connect the last vertex $u_{n'}$ of P' to v_n . Since $u_{n'}$ is hook visible by property (d), we connect it to v_n with an edge that starting from $u_{n'}$ reaches the access interval of $u_{n'}$, crosses the spine between the last vertex of γ' and p_2 and reaches p_1 from above. As shown in Fig. 3(c) the two edges (v_1, u_1) and $(u_{n'}, v_n)$ can be added without creating any crossing. Property (a)

holds by construction. About properties (b) and (c), we added two arcs in the bottom page. The first one connects a point immediately before v_n and a point immediately after it, so the segment of the spine between v_n and its following vertex is accessible from below; also, the addition of this arc does not change the accessibility of the spine crossing of γ' . The second arc added in the bottom page connects a point q in the access interval of $u_{n'}$ with a point immediately after u_1 ; by property (d) of γ' there is no vertex or spine crossings between q and u_1 . Thus the segments connecting u_1 to its preceding and to its following vertices are visible from below and property (b) holds; furthermore the addition of this arc does not change the accessibility to existing spine crossings. Since the new created spine crossings are visible from below, property (c) also holds. It is immediate to see that also property (d) holds; see for example Fig. 3(c).

Case 2: P and σ are not a minimally balanced pair. In this case there exists a prefix (i.e. a subpath) P' of P and a corresponding prefix σ' of σ that are balanced. P' is 2-colored path and σ' is a 2-colored sequence compatible with P' and their length is less than n . By induction, P' admits a topological book embedding γ' consistent with σ' and satisfying properties (a)–(d). On the other hand, $P'' = P \setminus P'$ is also a 2-colored path and $\sigma'' = \sigma \setminus \sigma'$ is a 2-colored sequence consistent with P'' . Thus, P'' also admits a topological book embedding γ'' consistent with σ'' and satisfying properties (a)–(d). Since the last vertex u_n of P' is hook visible in γ' and the first vertex w_1 of P'' is visible from above in γ'' , the two vertices can be connected with an edge that crosses the spine twice (see Fig. 3(d)), thus creating a topological book embedding γ of P consistent with σ . Property (a) holds by construction. The only arc added in the bottom page connects a point q in the access interval of u_n and a point q' immediately after the first vertex u_1 of P' . By property (d) of γ' there is no vertex or spine crossing between q and u_1 and between u_1 and q' , thus properties (b) and (c) hold for γ . Property (d) holds because it holds for γ' and γ'' . \square

Lemma 9. *A 3-colored path admits a topological book embedding with at most two spine crossings per edge consistent with any compatible 3-colored sequence.*

Proof (sketch). Let c_2 be a color distinct from the colors of the end-vertices of P . Let v_1, v_2, \dots, v_k be a maximal subpath of P colored c_2 . Let u_1 and u_2 be the vertices along P before v_1 and after v_k , respectively. We replace the subpath $u_1, v_1, v_2, \dots, v_k, u_2$ with an edge (u_1, u_2) . We do the same for every maximal subpath colored c_2 . Let P' be the resulting 2-colored path and σ' be the 2-colored sequence obtained from σ by removing all elements of color c_2 .

By Lemma 8, P' admits a topological book embedding γ' consistent with σ' that satisfies properties (a), (b), (c) and (d). We add to γ' a set Q of points colored c_2 to represent the removed vertices that will be added back. These points must be placed so that the sequence of colors along the spine coincides with σ . By property (b) of γ' all these points can be placed so that they are accessible from below. We now have to replace some edges of P' with paths of vertices colored c_2 . Let (u_1, u_2) be an edge that has to be replaced by a path $u_1, v_1, v_2, \dots, v_k, u_2$. For each vertex v_i to be added ($i = 1, 2, \dots, k$) we add an

image point to the drawing. The image points are added as follows. By property (a), the edge (u_1, u_2) crosses the spine at least once. Let χ be the point where (u_1, u_2) crosses the spine for the first time when going from u_1 to u_2 . By property (c) χ is visible from below. This means there is a segment s of ℓ with χ as an endpoint that is visible from below. We place $k - 1$ image points p_1, p_2, \dots, p_{k-1} inside this segment, while χ is the k -th image point p_k (it is the leftmost if s is to the left of χ , while it is the rightmost if s is to the right of χ). The first arc of the edge (u_1, u_2) is replaced by an arc connecting u_1 to p_1 . Each image point p_i is connected to the p_{i+1} ($i = 1, 2, k - 1$) by means of an arc in the top page. Finally, the last image point p_k is already connected to u_2 by means of the remaining part of the original edge (u_1, u_2) . Notice that the edge (u_1, p_1) does not cross the spine, and the same is true for any edge (p_i, p_{i+1}) , while the edge (p_k, u_2) crosses the spine at most once (the original edge had at most two spine crossing one of which was at $\chi = p_k$). We have replaced the edge (u_1, u_2) with a path $\pi = \langle u_1, p_1, p_2, \dots, p_k, u_2 \rangle$ with $k + 1$ edges, as needed. However, the points representing the intermediate vertices of this path are not the points of the set Q . The idea then is to “connect” the image points to the points of Q . To this aim, we add matching edges in the bottom page between the image points and the points of Q . Since both the points of Q and the image points are visible from below, these matching edges do not cross any other existing edge. Moreover, by using a simple brackets matching algorithm, we can add the matching edges so that they do not cross each other. Finally the matching edges can be used to create the actual path that represent $u_1, v_1, v_2, \dots, v_k, u_2$. \square

The following theorem is a consequence of Lemmas 2 and 9.

Theorem 5. *Every 3-colored path admits a 3-colored point-set embedding with curve complexity at most 5 on any compatible 3-colored point set.*

Theorem 5 can be extended to a subclass of 3-colored caterpillars.

Theorem 6. *Every 3-colored caterpillar with monochromatic leaves admits a 3-colored point-set embedding with curve complexity at most 5 on any compatible 3-colored point set.*

The above results motivate the study of 4-colored graphs, in particular a natural question is whether 4-colored paths admit point-set embedding on any set of points with constant curve complexity.

Theorem 7. *Let P be a 4-colored path with n vertices and let S be a 4-colored point set compatible with P . If the first $h \geq 2$ vertices along P only have two colors and the remaining $n - h$ only have the other two colors, then P has a 4-colored point-set embedding on S with curve complexity at most 5.*

5 Open Problems

Motivated by the results of this paper we suggest the following open problems:
 (i) Investigate whether the lower bound of Theorem 3 is tight. We recall that an

upper bound of $O(n)$ holds for all n -colored planar graphs [9]. (ii) Study whether constant curve complexity can always be guaranteed for 4-colored paths. (iii) Characterize the 3-colored caterpillars that admit a 3-colored point-set embedding with constant curve complexity on any given set of points.

References

1. Badent, M., Di Giacomo, E., Liotta, G.: Drawing colored graphs on colored points. *Theor. Comput. Sci.* **408**(2–3), 129–142 (2008)
2. Di Giacomo, E., Didimo, W., Liotta, G., Meijer, H., Trotta, F., Wismath, S.K.: k -colored point-set embeddability of outerplanar graphs. *J. Graph Algorithms Appl.* **12**(1), 29–49 (2008)
3. Di Giacomo, E., Gasieniec, L., Liotta, G., Navarra, A.: Colored point-set embeddings of acyclic graphs. CoRR 1708.09167 (2017), [arXiv:1708.09167](https://arxiv.org/abs/1708.09167)
4. Di Giacomo, E., Liotta, G.: The Hamiltonian augmentation problem and its applications to graph drawing. In: Rahman, M.S., Fujita, S. (eds.) WALCOM 2010. LNCS, vol. 5942, pp. 35–46. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11440-3_4
5. Di Giacomo, E., Liotta, G., Trotta, F.: On embedding a graph on two sets of points. *Int. J. Found. Comput. Sci.* **17**(5), 1071–1094 (2006)
6. Di Giacomo, E., Liotta, G., Trotta, F.: Drawing colored graphs with constrained vertex positions and few bends per edge. *Algorithmica* **57**(4), 796–818 (2010)
7. Frati, F., Glisse, M., Lenhart, W.J., Liotta, G., Mchedlidze, T., Nishat, R.I.: Point-set embeddability of 2-colored trees. In: Didimo, W., Patrignani, M. (eds.) GD 2012. LNCS, vol. 7704, pp. 291–302. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36763-2_26
8. Kaufmann, M., Wiese, R.: Embedding vertices at points: few bends suffice for planar graphs. *J. Graph Algorithms Appl.* **6**(1), 115–129 (2002)
9. Pach, J., Wenger, R.: Embedding planar graphs at fixed vertex locations. *Graphs Comb.* **17**(4), 717–728 (2001)

Planar Drawings of Fixed-Mobile Bigraphs

Michael A. Bekos¹, Felice De Luca², Walter Didimo^{2(✉)}, Tamara Mchedlidze³,
Martin Nöllenburg⁴, Antonios Symvonis⁵, and Ioannis G. Tollis⁶

¹ University of Tübingen, Tübingen, Germany
bekos@informatik.uni-tuebingen.de

² Università degli Studi di Perugia, Perugia, Italy
felice.deluca@studenti.unipg.it, walter.didimo@unipg.it

³ Karlsruhe Institute of Technology, Karlsruhe, Germany
mched@iti.uka.de

⁴ TU Wien, Vienna, Austria

noellenburg@ac.tuwien.ac.at

⁵ National Technical University of Athens, Athens, Greece
symvonis@math.ntua.gr

⁶ University of Crete, Heraklion, Greece
tollis@csd.uoc.gr

Abstract. A *fixed-mobile bigraph* G is a bipartite graph such that the vertices of one partition set are given with *fixed* positions in the plane and the *mobile* vertices of the other part, together with the edges, must be added to the drawing. We assume that G is planar and study the problem of finding, for a given $k \geq 0$, a planar poly-line drawing of G with at most k bends per edge. In the most general case, we show NP-hardness. For $k = 0$ and under additional constraints on the positions of the fixed or mobile vertices, we either prove that the problem is polynomial-time solvable or prove that it belongs to NP. Finally, we present a polynomial-time testing algorithm for a certain type of “layered” 1-bend drawings.

1 Introduction

This paper considers the following problem. Let $G = (V_f, V_m, E)$ be a planar bipartite graph such that the vertices in V_f , called *fixed vertices*, have fixed distinct locations (points) in the plane, while the vertices in V_m , called *mobile vertices*, can be freely placed. Does G admit a crossing-free drawing Γ with at most k bends per edge, where k is a given non-negative integer? We assume that each vertex of G is drawn in Γ as a distinct point of the plane and that each edge is drawn as a simple poly-line. We call G an *FM-bigraph* and a drawing Γ with the properties mentioned above a *planar k -bend drawing* of G . In particular, since edge bends negatively affect the readability of a graph layout (see, e.g., [32, 33]),

Research in this work started at the Bertinoro Workshop on Graph Drawing 2016. We thank all the participants and in particular S.-H. Hong for useful discussions. We also thank an anonymous reviewer of this research for some valuable comments, and especially for suggesting the idea behind the proof of Theorem 6.

we are mainly interested in drawings with small values of k , ideally with $k = 0$ (i.e., straight-line drawings). We define the *bend number* of G as the minimum value of k for which G admits a planar k -bend drawing.

Besides its intrinsic theoretical interest, our problem is motivated by the following practical scenario. Fixed vertices represent geographic locations and each mobile vertex is an attribute of one or more locations. One wants to place each mobile vertex in the plane and connect it to its associated locations, while guaranteeing a “readable” layout. We interpret readability in terms of planarity and small number of bends per edge. Other criteria can also be studied, like for example angular resolution, edge length, drawing area (see, e.g., [10, 34]).

Contribution. We introduce k -bend drawings of FM-bigraphs with a focus on $k = 0$ and $k = 1$. Our results are as follows:

(i) We prove that computing the bend number of an FM-bigraph G is NP-hard. More generally, deciding whether G admits a planar k -bend drawing is at least as hard as deciding whether an n -vertex graph admits a planar embedding with given correspondence (mapping) on a set of n points, such that each edge has at most $2k + 1$ bends (Sect. 2.1). If all fixed vertices of G are collinear, the existence of a 0-bend drawing can be tested in linear time.

(ii) Since it is difficult to discretize the problem in the general case [15, 30], we investigate the case in which each mobile vertex is restricted to lie in the convex hull of its neighbors. This scenario is reasonable in practice, as the user may expect that each attribute is placed in a sort of “barycentric” position with respect to its associated locations. In this setting, we prove that testing the existence of a 0-bend drawing is a problem in NP. With a reduction to a combinatorial problem, which is of its own independent interest (but unfortunately NP-hard in its general form), we obtain polynomial-time solutions when the intersection graph of the convex hulls is a path, a cycle or, more generally, a cactus (Sect. 2.2).

(iii) We finally study 1-bend drawings of FM-bigraphs in a convention called *h-strip drawing model*, inspired by practical labeling scenarios [3]. All fixed vertices are partitioned into a finite set of horizontal strips and each mobile vertex is placed outside these strips. Edges are not allowed to cross any strip, i.e., to intersect both its top and bottom side; see also Fig. 4. For this model we provide polynomial-time testing algorithms (Sect. 3).

Related Work. Our problem is related to several problems addressed in the literature, but it also has substantial differences from all of them.

Point labeling. A close connection is with the problem of labeling a given set of points in the plane (see, e.g., [28, 37]), because mobile vertices can be regarded as labels for the fixed vertices (points). Similarly to our setting, the *many-to-one boundary labeling* problem [3, 23] assumes that each label can have multiple associated vertices and it is visually connected to them by poly-line edges. However, edges can only be drawn as chains of horizontal and vertical segments (which may partially overlap), and the labels are placed outside a single rectangular region that encloses all vertices. Variants of the boundary labeling problem, where each fixed vertex is associated with exactly one label are also studied in the literature

(see, e.g., [2, 5, 22]). Note that, in labeling problems, labels are geometric shapes of non-empty area, while we model mobile vertices as points.

Partial drawings. Our problem is a special case of the problem of extending a partial drawing of a (not necessarily bipartite) planar graph G to a planar straight-line drawing of G . This problem is NP-hard in general [30] and polynomial-time solvable for restricted cases [9, 13, 18, 24, 36].

Point-set embedding. In a point-set embedding problem, a planar graph with n vertices must be planarly mapped onto a given set of n points, with or without a predefined correspondence between the vertices and the points (see, e.g., [1, 8, 11, 12, 21, 29]). Thus, in all settings of the point-set embedding problem, each vertex can only be mapped to a finite set of points. The results in [1, 29] imply that any n -vertex planar FM-bigraph admits a k -bend drawing with $k = O(n)$. Indeed, [1, 29] prove that any n -vertex planar graph can be planarly mapped onto any set of n points, with given correspondences, using a linear number of bends per edge (which is also necessary in some cases). Hence, for a given FM-bigraph, one can place the mobile vertices anywhere so to realize a planar drawing.

Constrained drawings of bipartite graphs. Misue [26] proposed a model and a technique for drawing bipartite graphs such that the vertices of a partition set, called *anchors*, are evenly distributed on a circle. Anchors are similar to fixed vertices in our setting, but the order of the anchors in Misue's model can be freely chosen. Extensions to the 3D space and to semi-bipartite graphs have been subsequently presented [19, 27]. Finally, several papers study how to draw a bipartite graph such that the vertices of each partition set are on a line or within a specific plane region (see, e.g., [6, 7, 14]). In these scenarios, the vertices do not have predefined locations.

Notation. We assume familiarity with graph theory (see, e.g., [17]). For standard definitions on *planar graphs* and *drawings*, we point the reader to [20, 35].

We denote an FM-bigraph by a pair $\langle G, \phi \rangle$, where $G = (V_f, V_m, E)$ is a bipartite graph and $\phi : V_f \rightarrow \mathbb{R}^2$ is a function that maps each vertex $v \in V_f$ to a distinct point $p_v = \phi(v)$. A k -bend drawing of $\langle G, \phi \rangle$ is a k -bend drawing of G such that each vertex $v \in V_f$ is mapped to $\phi(v)$. In order to study the complexity of computing the bend number of planar FM-bigraphs, we introduce the k -BEND FM-BIGRAPH decision problem: *Given a planar FM-bigraph $\langle G, \phi \rangle$ and a non-negative integer k , is there a planar k -bend drawing of $\langle G, \phi \rangle$?*

From now on, we assume that G is planar. Also, we let $n_f = |V_f|$, $n_m = |V_m|$, and $n = n_f + n_m$. For reasons of space, some proofs are sketched or omitted; full proofs can be found in [4].

2 Straight-Line Planar Drawings of FM-Bigraphs

We show that the 0-BEND FM-BIGRAPH problem is NP-hard (Theorem 1), which implies that it is NP-hard to compute the bend number of a planar FM-bigraph. A simple linear-time testing algorithm is given when all fixed vertices are collinear (Theorem 3). If each mobile vertex must be placed inside the convex

hull of its neighbors, then the 0-BEND FM-BIGRAPH problem belongs to NP (Theorem 4), and it becomes polynomial-time solvable if the intersection graph of the convex hulls is a cactus (Theorem 5).

2.1 NP-Hardness and Collinear Fixed Vertices

To prove that 0-BEND FM-BIGRAPH is NP-hard we use a reduction from the 1-bend point set embeddability with correspondence problem (or 1-BPSEWC, for short), which has been proven to be NP-hard by Goaoc *et al.* [15]. Problem 1-BPSEWC is defined as follows: *Given a planar graph $G = (V, E)$, a set S of $|V|$ points in the plane, and a one-to-one correspondence ζ between V and S , is there a planar 1-bend drawing of G such that each vertex v is mapped to point $\zeta(v)$?*

Theorem 1. *The 0-BEND FM-BIGRAPH problem is NP-hard, even if each mobile vertex has degree at most two.*

Proof. Let $\langle G = (V, E), S, \zeta \rangle$ be an instance of 1-BPSEWC. Construct (in linear time) an instance $\langle G' = (V_f, V_m, E'), \phi \rangle$ of 0-BEND FM-BIGRAPH as follows: Let $V_f = V$ and $\phi = \zeta$; for each edge $e = (u, v) \in E$, define a corresponding vertex $w_e \in V_m$ and two edges $(w_e, u), (w_e, v)$ in E' . Clearly, G has a 1-bend drawing Γ that respects ζ if and only if G' has a planar 0-bend drawing Γ' that respects ϕ : The position of a bend along an edge $e = (u, v)$ of Γ corresponds to the positions of the mobile vertex w_e in Γ' ; if e has no bend, w_e is drawn anywhere along segment \overline{uv} . \square

The reduction in Theorem 1 can be applied with no change to prove that, for any $k \geq 0$, problem k -BEND FM-BIGRAPH is at least as difficult as problem $(2k + 1)$ -BPSEWC, which allows up to $(2k + 1)$ bends per edge.

Theorem 2. *The k -BEND FM-BIGRAPH problem is at least as hard as the $(2k+1)$ -BPSEWC problem, for any $k \geq 0$.*

When all fixed vertices of an FM-bigraph $\langle G, \phi \rangle$ are collinear, it can be checked in linear time whether $\langle G, \phi \rangle$ admits a planar 0-bend drawing.

Theorem 3. *Let $\langle G = (V_f, V_m, E), \phi \rangle$ be an n -vertex FM-bigraph such that all vertices of V_f are collinear. There exists an $O(n)$ -time algorithm that tests whether $\langle G, \phi \rangle$ admits a planar 0-bend drawing.*

Proof (sketch). Let ℓ be the line passing through all fixed vertices. Deciding whether $\langle G, \phi \rangle$ has a planar 0-bend drawing coincides with testing the planarity of a graph obtained by augmenting G with a cycle that connects all fixed vertices in the order they appear along ℓ . \square

2.2 Mobile Vertices at Internal Positions

We now focus on *convex-hull drawings*, in which all fixed vertices are in general position and each vertex $u_m \in V_m$ lies in the convex hull of its neighbors. With

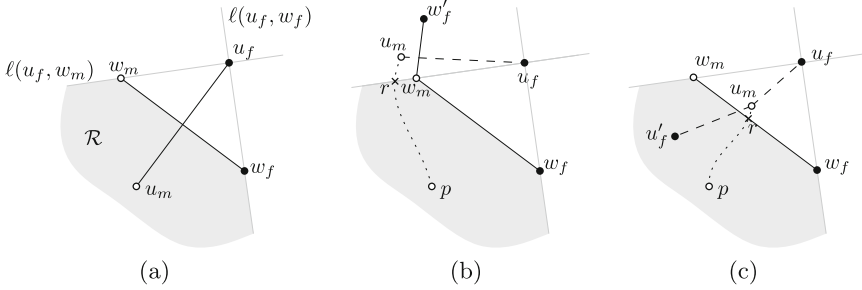


Fig. 1. Illustration of the proof of Lemma 1. (b-c) Part of trajectory T between r and r' is shown by a dotted line.

slight abuse of notation, we denote by $CH(u_m)$ the convex hull of the neighbors of u_m . Let $\mathcal{A} = \mathcal{A}(V_f)$ be the arrangement of lines defined by all pairs of fixed points; see Fig. 2(a). \mathcal{A} has $O(n_f^2)$ lines and $O(n_f^4)$ cells [16]. Lemma 1 allows us to discretize the set of possible positions for the mobile vertices; it implies that all positions of u_m in the same cell of \mathcal{A} within $CH(u_m)$ are equivalent for a planar 0-bend drawing of $\langle G, \phi \rangle$.

Lemma 1. *Let $\langle G = (V_f, V_m, E), \phi \rangle$ be an FM-bigraph, $u_m \in V_m$, and C a cell of $\mathcal{A} = \mathcal{A}(V_f)$ inside $CH(u_m)$. Let also p and p' be two points in C . Suppose that Γ is a 0-bend drawing of $\langle G, \phi \rangle$ where u_m is at point p and let Γ' be a 0-bend drawing of $\langle G, \phi \rangle$ obtained from Γ by only moving u_m from point p to point p' . Then Γ' is planar if and only if Γ is planar.*

Proof. Suppose by contradiction that Γ' is planar and Γ is not (the proof for the other direction is symmetric). This implies that while moving u_m along some trajectory T from p' to p inside cell C , at some point we get a crossing along one of the edges incident to u_m . Let r be the point of T closest to p' , such that placing u_m at r causes such a crossing (i.e., placing u_m on any point between r and p' implies no crossing). Let (u_m, u_f) be an edge crossed by some other edge (w_m, w_f) , when u_m is placed at r . Assume w.l.o.g. that $w_m \in V_m$ and $w_f \in V_f$, and that u_m lies to the right of the oriented edge (w_m, w_f) ; see Fig. 1(a). Denote by $\ell(u_f, w_f)$ the line through u_f and w_f and by $\ell(u_f, w_m)$ the line through u_f and w_m . Let \mathcal{R} be the region delimited by lines $\ell(u_f, w_f)$, $\ell(u_f, w_m)$ and edge (w_m, w_f) that contains u_m . Let r' be a point of T lying between r and p' . Notice that r' has to lie outside of \mathcal{R} . Thus T crosses the border of \mathcal{R} . Let us additionally assume that r' lies arbitrarily close to the border of \mathcal{R} . We distinguish three cases, based on whether T crosses $\ell(u_f, w_f)$, $\ell(u_f, w_m)$, or edge (w_m, w_f) .

Case 1. T crosses $\ell(u_f, w_f)$. Since line $\ell(u_f, w_f)$ is part of \mathcal{A} , r' lies outside C . This is a contradiction to the assumption that T lies in C .

Case 2. T crosses $\ell(u_f, w_m)$; see Fig. 1(b). Since w_m is in the convex hull of its neighbors, there is an edge (w_m, w'_f) with w_f and w'_f on different sides of

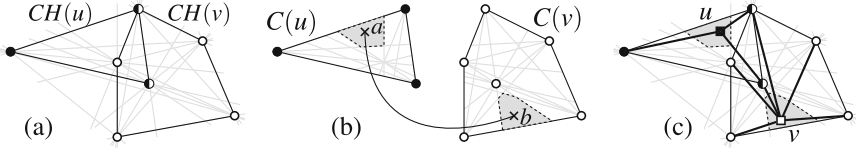


Fig. 2. (a) Line arrangement $\mathcal{A} = \mathcal{A}(V_f)$ with the neighbors $N(u)$ in black and $N(v)$ in white; $CH(u)$ and $CH(v)$ intersect and thus form an edge in G_x . (b) Two clusters $C(u)$ and $C(v)$ of G_c with one exemplary edge between two cell vertices a and b . (c) Placing u inside cell(a) and v inside cell(b) yields a planar drawing of the FM-bigraph (thick edges).

$\ell(u_f, w_m)$ (if not, the crossing is resolved). Placing u_m at r' yields a crossing with (w_m, w'_f) , as r' is arbitrarily close to $\ell(u_f, w_m)$; a contradiction to the choice of r .

Case 3. T crosses (w_m, w_f) . Since u_m lies in the convex hull of its neighbors, there is an edge (u_m, u'_f) , where u'_f and u_f lie on different sides of the line through edge (w_f, w_m) ; see Fig. 1(c). Placing u_m at r' would introduce a crossing between (u_m, u'_f) and (w_m, w_f) , as r' lies arbitrarily close to (w_m, w_f) . This again contradicts the choice of r . \square

Lemma 1 implies that, the 0-BEND FM-BIGRAPH problem belongs to NP for convex-hull drawings¹. A non-deterministic algorithm guesses an assignments of the mobile vertices to the $O(n_f^4)$ cells and, since G is planar, checks in $O(n_f^2)$ time whether the corresponding 0-bend drawing is planar (note that $n_m = O(n_f)$). We summarize this observation in the following theorem.

Theorem 4. *The 0-BEND FM-BIGRAPH problem belongs to NP if each mobile vertex must lie in the convex hull of its neighbors.*

Central ingredients to prove that the problem is in fact in P for certain input configurations are the CH intersection graph G_x of G , the cell graph G_c of G , and the skeleton graph G_s of G_c , which we formally define in the following.

The *CH intersection graph* G_x is defined as the intersection graph [25] of the convex hulls $CH(u)$ over all $u \in V_m$.

The *cell graph* G_c is a clustered graph defined as follows; see Fig. 2(b) for an example. Each mobile vertex u is associated with a cluster $C(u)$; the vertices of $C(u)$, called *cell vertices*, are the cells of \mathcal{A} that intersect with $CH(u)$ (and in fact are contained in $CH(u)$). The vertices of G_c are defined by the disjoint union of the vertices of all clusters², that is, $V(G_c) = \uplus_{u \in V_m} C(u)$. For a cell vertex a of G_c , we denote by $\text{cell}(a)$ the cell corresponding to a in \mathcal{A} . For a pair of mobile vertices u and v such that $CH(u) \cap CH(v) \neq \emptyset$, a cell vertex $a \in C(u)$

¹ We remark that in a preliminary version of [30], it is claimed membership in NP for the partial planarity extension problem [31], which would imply membership in NP also for our problem. That claim, however, lacks a proof in [31] and the author was only able to prove the NP-hardness of the problem in [30] (personal communication).

² Cells in the intersection of two convex hulls correspond to different vertices of G_c .

is adjacent to a cell vertex $b \in C(v)$ if and only if placing u in $C(a)$ and v in $C(b)$ produces no crossing among the edges incident to u and v ; see Fig. 2(c). Note that G_c has $O(n_f^4 n_m)$ vertices and $O(n_f^8 n_m^2)$ edges. Also, by definition, for each pair of mobile vertices u and v such that $CH(u) \cap CH(v) \neq \emptyset$, u and v can be positioned within their convex hulls without creating edge crossings if and only if there exist two adjacent cell vertices $a \in C(u)$ and $b \in C(v)$ in G_c .

The *skeleton graph* G_s is created by selecting exactly one cell vertex, called a *skeleton vertex*, from each cluster of G_c , such that for every pair of mobile vertices u and v with $CH(u) \cap CH(v) \neq \emptyset$, the skeleton vertices of $C(u)$ and $C(v)$ are adjacent in G_c . Graph G_s is the subgraph of G_c induced by the skeleton vertices. Note that G_s might not exist. If G_s exists, then it is isomorphic to G_x . The following characterization is an immediate consequence of our definitions.

Lemma 2. *An FM-bigraph $\langle G, \phi \rangle$ admits a planar 0-bend convex-hull drawing if and only if cell graph G_c has a skeleton.*

Proof. A planar 0-bend drawing immediately defines a skeleton. Conversely, if G_c has a skeleton G_s , a planar 0-bend drawing Γ is obtained by placing each $u \in V_m$ in the cell corresponding to the skeleton vertex of $C(u)$ in G_s . Since crossings may only occur between edges incident to mobile vertices u and v such that $CH(u) \cap CH(v) \neq \emptyset$, Γ is planar. □

The characterization of Lemma 2 allows us to translate the geometric problem of finding a 0-bend convex-hull drawing of an FM-bigraph bigraph $\langle G, \phi \rangle$ to a purely combinatorial problem on a support clustered graph G_c constructed from $\langle G, \phi \rangle$. Unfortunately, however, this combinatorial problem is NP-hard in its general form, as Theorem 6 shows. Nonetheless, we are able to solve it efficiently when G_x is a cactus (Theorem 5), which includes the special cases in which G_x is a cycle or a tree. The next two lemmas are base cases for Theorem 5.

Lemma 3. *Let $\langle G, \phi \rangle$ be an FM-bigraph such that G_x is a path. There exists a polynomial-time algorithm that tests whether $\langle G, \phi \rangle$ has a planar 0-bend convex-hull drawing.*

Proof. By Lemma 2, it is enough to test whether G_c has a skeleton. Let u_1, \dots, u_λ be the mobile vertices in the order their convex hulls appear along path G_x . Call a cell vertex a of $C(u_i)$ *active* if and only if the subgraph of G_c induced by $C(u_1) \cup \dots \cup C(u_i)$ has a skeleton containing a , where $1 \leq i \leq \lambda$. Thus, G_c has a skeleton if and only if there is an active cell vertex in $C(u_\lambda)$. A simple algorithm that tests this condition works as follows. Initially mark all cell vertices of $C(u_1)$ as active, and then *propagate this information forward* to the cell vertices of $C(u_\lambda)$, that is, for each $i = 2, \dots, \lambda$, mark each cell vertex of $C(u_i)$ as active if it has an active neighbor in $C(u_{i-1})$. The time complexity is bounded by the number of vertices and edges in G_c . □

Lemma 4. *Let $\langle G, \phi \rangle$ be an FM-bigraph such that G_x is a simple cycle. There exists a polynomial-time algorithm that tests whether $\langle G, \phi \rangle$ has a planar 0-bend convex-hull drawing.*

Proof. Let u_1, \dots, u_λ be the mobile vertices in the cyclic order their convex hulls appear along G_x . Our approach is similar to the one of Lemma 3, but now it is not enough to propagate the information about the active vertices only from $C(u_1)$ to $C(u_\lambda)$. Indeed, there might be vertices of $C(u_1)$ that cannot close a cycle with an active vertex of $C(u_\lambda)$.

In the first phase, our refined algorithm starts by marking all cell vertices of $C(u_1)$ as active and then propagates this information forward to $C(u_\lambda)$, as in the case of a path. However, all cell vertices of a cluster that are not marked as active at the end of this phase are now definitely removed from G_c (along with their incident edges), as they cannot occur in any skeleton. Then, the algorithm cleans all vertex marks and executes a *backward propagation* phase from $C(u_\lambda)$ to $C(u_1)$ (symmetric to the previous one), where all the remaining vertices in $C(u_\lambda)$ are initially marked as active. As before, all vertices that are not marked as active at the end of this phase are definitely removed from G_c . Now, the algorithm removes from $C(u_1)$ all vertices with no neighbor in $C(u_\lambda)$ and from $C(u_\lambda)$ all vertices with no neighbor in $C(u_1)$, as these vertices cannot occur in a skeleton of G_c . Finally, for each pair of adjacent vertices $v \in C(u_1)$ and $w \in C(u_\lambda)$ in G_c , the algorithm checks whether there exists a path π_{vw} from v to w that passes through each $C(u_j)$ exactly once ($j = 1, \dots, \lambda$). If π_{vw} exists, both v and w are marked as *confirmed*. At the end, every vertex in $C(u_1) \cup C(u_\lambda)$ that is not confirmed is removed from G_c , as it cannot occur in a skeleton. Conversely, by construction, every remaining vertex v in $C(u_1)$ has an adjacent vertex $w \in C(u_\lambda)$ such that $\pi_{vw} \cup (v, w)$ is a skeleton (simple cycle) of G_c . Thus the test is positive if and only if $C(u_1)$ is not empty.

It is immediate to see that the whole testing algorithm works in polynomial time in the size of G_c and, if the test is positive, a skeleton of G_c can be easily reconstructed visiting G_c from any vertex $v \in C(u_1)$. \square

We now extend the previous result to the case that G_x is a cactus, which also covers the case of a tree. We recall that a cactus is a connected graph in which any two simple cycles share at most one vertex. A cactus is an outerplanar graph and can always be decomposed into a tree where each node corresponds to either a single vertex or a simple cycle (refer to Fig. 3(a)).

Theorem 5. *Let $\langle G, \phi \rangle$ be an FM-bigraph such that G_x is a cactus. There exists a polynomial-time algorithm that tests whether $\langle G, \phi \rangle$ has a planar 0-bend convex-hull drawing.*

Proof. By Lemmas 3 and 4, the statement holds when G_x is a path or a cycle. In the general case, our testing algorithm decomposes G_x into its tree \mathcal{T} (as in Fig. 3(b)), roots \mathcal{T} at any node, and visits \mathcal{T} bottom-up. More precisely, each vertex of G_x corresponds to a convex hull $CH(u)$ of a mobile vertex u , and it has a one-to-one correspondence with a cluster $C(u)$ of G_c . Thus, each node μ of \mathcal{T} corresponds to either a single cluster of G_c or to a cycle of clusters of G_c . Note that when in G_x two cycles share a vertex (cluster of G_c), we replicate such a vertex in both nodes of \mathcal{T} that correspond to the two cycles. For example, in Fig. 3(b) cluster C_{11} inside μ_6 and cluster C_3 inside μ_1 correspond to the same

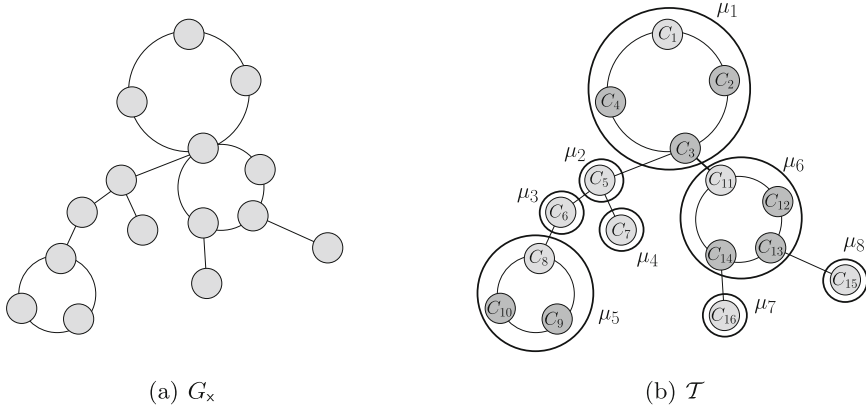


Fig. 3. (a) An intersection graph G_x that is a cactus. (b) The decomposition tree \mathcal{T} of G_x . Clusters C_{11} of μ_6 and C_3 of μ_1 correspond to the same vertex of G_x .

vertex of G_x . Once the root of \mathcal{T} has been chosen, we define the *anchor* of μ as the cluster that connects μ to its parent node in \mathcal{T} (light-gray in Fig. 3(b)). During the bottom-up visit of \mathcal{T} , two cases are possible when a node μ is visited:

- **μ is a leaf.** If μ contains a single cluster (i.e., its anchor), then all its cell vertices are marked as active; if μ contains a cycle of clusters, then the active cell vertices of its anchor are computed as for $C(u_1)$ in the proof of Lemma 4.
- **μ is an internal node.** Let $\nu_1, \nu_2, \dots, \nu_k$ be the children of μ in \mathcal{T} and denote by C_{q_i} the cluster of μ connected to the anchor of ν_i . Note that C_{q_i} may coincide with some C_{q_j} if $q_i \neq q_j$. Also, the anchor of ν_i and C_{q_i} may correspond to the same vertex of G_x .
 - For each $i = 1, \dots, k$, if the anchor of ν_i differs from C_{q_i} in G_x , remove from C_{q_i} all cell vertices that are not connected to an active cell vertex of the anchor of ν_i in G_c , as they cannot occur in any skeleton of G_c . On the other hand, if the anchor of ν_i and C_{q_i} coincide in G_x , remove from C_{q_i} all vertices of the anchor of ν_i that are not marked as active.
 - Now, if μ contains a single cluster (i.e., its anchor), then all its remaining cell vertices are marked as active; if μ contains a cycle of clusters, then the active cell vertices of its anchor are computed as in Lemma 4. At this point, if the anchor of μ contains no active vertex, the algorithm stops and the instance is rejected, as a skeleton does not exist.

Once the bottom-up visit of \mathcal{T} ends, the test is positive if and only if the anchor of the root node of \mathcal{T} has an active cell vertex w , and in this case one can reconstruct a skeleton of G_c starting from w and visiting \mathcal{T} top-down. In particular, during the top-down visit, for each node μ of \mathcal{T} , any active vertex in the anchor of μ can be arbitrarily selected, as it is connected to the parent node of μ by construction. Also, if μ corresponds to a simple cycle of clusters, the construction of a cycle that connects these clusters is done as in Lemma 4.

Concerning the time complexity, the above algorithm takes polynomial time in the size of G_c . Indeed, the number of clusters that may occur in multiple nodes of \mathcal{T} (i.e., those that are shared by multiple cycles of clusters) is at most the number of cycles in G_x . Therefore, the total number of clusters over all nodes of \mathcal{T} is linear in the number of clusters of G_c . This also implies that the total number of cell vertices over all clusters of \mathcal{T} is linear in the number of cell vertices in G_c . Finally, each node μ of \mathcal{T} is visited twice (once in the bottom-up visit and once in the top-down visit), and in each visit of μ the algorithm has a running time that is polynomial in the number of cell vertices in the clusters of μ . \square

Finally, we show the following NP-completeness result on a combinatorial generalization of our problem. We re-use the terminology of the 0-BEND FM-BIGRAPH problem to emphasize the analogies.

Theorem 6. *Let $G_x = (\mathcal{C}, \mathcal{E})$ be a graph, where \mathcal{C} is a set of disjoint clusters of cells. Also, let $G_c = (V, E)$ be a graph, where each $v \in V$ is a cell of a cluster $C(v) \in \mathcal{C}$ and $(u, v) \in E$ only if $(C(u), C(v)) \in \mathcal{E}$. It is NP-complete to test if there is a subset $V' \subseteq V$ of skeleton vertices, containing exactly one cell from each cluster in \mathcal{C} such that the induced subgraph $G_c[V']$ is isomorphic to G_x .*

Proof (sketch). The problem is clearly in NP. The hardness proof is by reduction from 3SAT. For a boolean 3SAT formula ψ create a cluster $C(x)$ for each variable x in ψ and a cluster $C(\gamma)$ for each clause γ of ψ . In G_x each clause cluster is adjacent to the three clusters of the variables occurring in the clause. Each variable cluster $C(x)$ consists of two cells in G_c , one for the positive literal x and one for its negation $\neg x$. Also, each clause cluster $C(\gamma)$ contains three cells, one for each literal. Finally, connect each literal cell of a clause λ to the corresponding cell of its variable cluster and to all four cells of the other two variables of γ . It can be seen that ψ has a satisfying truth assignment iff there exists a subset of skeleton vertices in G_c that induces a subgraph isomorphic to G_x . \square

3 1-Bend Drawings in the h -Strip Drawing Model

Our model for 1-BEND FM-BIGRAPHS is inspired by the boundary labeling approach [3], where mobile vertices are regarded as labels that must be connected to the fixed vertices. In the boundary labeling model, the fixed vertices are inside a single rectangular region and each label is either to the left or to the right of this region. Our model allows for multiple rectangular regions (corresponding to horizontal strips); each mobile vertex is placed outside of these regions, either below or above each of them. To avoid long edges and make the drawing more readable, edges are not allowed to traverse regions.

More formally, our model is called the *h-strip model* and is defined as follows. Let $\langle G = (V_f, V_m, E), \phi \rangle$ be an FM-bigraph and assume that the vertices of V_f all have distinct x -coordinates (this condition is always achievable by a suitable rotation of the plane). For the sake of simplicity, for a vertex $u \in V_f$, we do not distinguish between u and its fixed position $\phi(u)$. Let $\mathcal{S} = \{S_1, S_2, \dots, S_h\}$

($h \geq 1$) be a top-to-bottom sequence of (closed) disjoint horizontal strips of the plane that partition V_f , i.e., each S_i has a finite height and infinite width, each vertex of V_f lies in one S_i , and $S_i \cap S_{i+1} = \emptyset$ for $i = 1, \dots, h - 1$. Since the strips are disjoint, there is always a non-empty region of the plane between two consecutive strips, which does not contain fixed vertices. Also, there are no fixed vertex above S_1 and below S_h . Any point that is not inside a strip is called a *free point*. For a vertex $u \in V_f$, the strip that contains u is called the *strip of u* .

A 1-bend drawing of G within \mathcal{S} is defined as follows; see Fig. 4: (i) Each vertex $v \in V_m$ is mapped to a distinct free point. (ii) Each edge $e = (u, v)$, with $u \in V_f$ and $v \in V_m$ consists of a segment \overline{vp} from v to a point p on the boundary of the strip of u and of a vertical segment \overline{pu} ; all points of \overline{vp} but p are free points, while \overline{pu} is completely inside the strip of u . (iii) No edge intersects the boundary of a strip twice and no two edges cross in a free point.

Note that in the h -strip model two distinct edges $e_1 = (u, v_1)$ and $e_2 = (u, v_2)$, where $u \in V_f$, share their vertical segments if these segments are incident to u both from below or both from above. This overlap does not create ambiguity and reduces the visual complexity caused by the edges. Figure 4 shows a 1-bend drawing of a bigraph within a given set of three strips (gray regions), with fixed vertices in black. Also note that, if an FM-bigraph G has no 1-bend drawing for a set \mathcal{S} of strips, splitting an element of \mathcal{S} into two strips may lead to a feasible solution; see Fig. 5(a). Conversely, splitting a strip may transform a positive instance into a negative one; see Fig. 5(b). We prove the following.

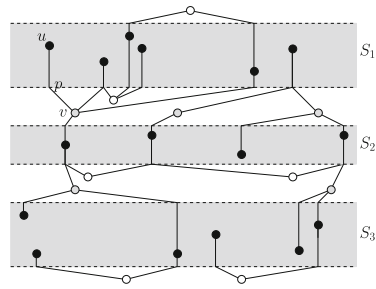


Fig. 4. A 3-strip drawing.

Theorem 7. *Let $\langle G = (V_f, V_m, E), \phi \rangle$ be an n -vertex FM-bigraph and let \mathcal{S} be a set of horizontal strips that partition V_f . There exists an $O(n)$ -time algorithm that tests whether $\langle G, \phi \rangle$ admits a 1-bend drawing within \mathcal{S} .*

Proof (sketch). Call a fixed vertex *black*, a mobile vertex with all neighbors in the same strip *white*, and the remaining vertices *gray*. A gray vertex with neighbors in two consecutive strips must lie between them, while each white vertex can lie either above or below the strip of its neighbors. If a grey vertex has neighbors that are not in two consecutive strips, the instance is immediately rejected.

Let $\mathcal{S} = \{S_1, \dots, S_h\}$ be the sequence of strips. For each strip $S_i \in \mathcal{S}$, let $V_f^i = \{u_1^i, \dots, u_{r_i}^i\}$ be the left-to-right sequence of black vertices inside S_i . Also, let V_m^i be the set of mobile vertices connected to some vertex of V_f^i . Arranging the vertices of V_m^i above or below S_i so to avoid crossings between their incident edges equals to assigning each of them either above or below the half-plane determined by a horizontal line that contains $u_1^i, \dots, u_{r_i}^i$, in this left-to-right order. Hence, testing if a 1-bend drawing within \mathcal{S} exists generalizes testing the existence of a 0-bend drawing when all fixed vertices are collinear. As in



Fig. 5. (a) An instance with a single strip and no solution (left); splitting the strip into two strips, the instance becomes feasible (right). (b) A positive instance with a single strip; splitting the strip into two strips, it becomes unfeasible.

Theorem 3, this problem is reduced to testing planarity of a graph G' suitably defined by augmenting G . Namely, for each S_i , add a cycle C_i connecting all edges of V_f^i in their left-to-right order; then, subdivide edge (u_l^i, u_r^i) of C_i with three vertices v_1^i, v_2^i, v_3^i , in this order from u_l^i to u_r^i , and call C'_i the subdivision of C_i ; finally, for each $i = 1, \dots, h-1$ and $j = 1, 2, 3$, connect v_j^i to v_j^{i+1} . Graph G' is planar iff it has a planar embedding where C'_i is inside C'_{i+1} ($i \in \{1, \dots, h-1\}$). A mobile vertex w between C'_i and C'_{i+1} corresponds to placing w above S_{i+1} and below S_i . If w is in the outer face of G' then w is below S_h , and if w is inside C'_1 then w is above S_1 . Since the size of G' is linear in the size of G and graph planarity testing is linear-time solvable, the statement holds. \square

The next result immediately follows by iterating the technique in the proof of Theorem 7 over all possible ways of partitioning V_f into h strips.

Corollary 1. *Let $\langle G = (V_f, V_m, E), \phi \rangle$ be an n -vertex FM-bigraph and let $h \in \mathbb{N}^+$ be a constant. There is an $O(|V_f|^{h-1}n)$ -time algorithm that tests if $\langle G, \phi \rangle$ has a 1-bend drawing within \mathcal{S} , for some set \mathcal{S} of h strips that partition V_f .*

4 Conclusions and Open Problems

We introduced FM-bigraphs, showed that the k -BEND FM-BIGRAPH problem is NP-hard in the general case, and gave polynomial-time algorithms for $k \in \{0, 1\}$ in some interesting restricted cases. Several open research questions remain:

Q1. We could solve the 0-BEND FM-BIGRAPH problem for convex-hull drawings if the CH intersection graph is a cactus and show that it is NP-complete in a non-geometric setting. Can we solve the problem for larger classes of convex-hull drawings in polynomial time or extend the NP-completeness to our geometric setting? More generally, for which other layout constraints or sub-families of FM-bigraphs does the k -BEND FM-BIGRAPH problem become tractable?

Q2. Our focus was on proving the existence of polynomial-time algorithms under certain layout constraints, but some of the algorithms have high time complexity. Thus, finding more efficient algorithms is of interest.

Q3. We focused on crossing-free drawings of FM-bigraphs. Relaxing the planarity requirement (e.g., for a given maximum number of permitted crossings per edge) is an interesting variant, as well as, designing heuristics or exact approaches for crossing/bend minimization.

References

1. Badent, M., Di Giacomo, E., Liotta, G.: Drawing colored graphs on colored points. *Theoret. Comput. Sci.* **408**(2–3), 129–142 (2008)
2. Barth, L., Gemsa, A., Niedermann, B., Nöllenburg, M.: On the readability of boundary labeling. In: Di Giacomo, E., Lubiw, A. (eds.) GD 2015. LNCS, vol. 9411, pp. 515–527. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27261-0_42
3. Bekos, M.A., Cornelsen, S., Fink, M., Hong, S., Kaufmann, M., Nöllenburg, M., Rutter, I., Symvonis, A.: Many-to-one boundary labeling with backbones. *J. Graph Algorithms Appl.* **19**(3), 779–816 (2015)
4. Bekos, M.A., De Luca, F., Didimo, W., Mchedlidze, T., Nöllenburg, M., Symvonis, A., Tollis, I.: Planar drawings of fixed-mobile bigraphs. CoRR 1708.09238 (2017)
5. Bekos, M.A., Kaufmann, M., Symvonis, A., Wolff, A.: Boundary labeling: models and efficient algorithms for rectangular maps. *Comput. Geom.* **36**(3), 215–236 (2007)
6. Biedl, T.C.: Drawing planar partitions I: LL-drawings and LH-drawings. In: Janardan, R. (ed.) *Computational Geometry (SoCG 1998)*, pp. 287–296. ACM (1998)
7. Biedl, T., Kaufmann, M., Mutzel, P.: Drawing planar partitions II: HH-drawings. In: Hromkovič, J., Sýkora, O. (eds.) WG 1998. LNCS, vol. 1517, pp. 124–136. Springer, Heidelberg (1998). https://doi.org/10.1007/10692760_11
8. Brandes, U., Erten, C., Estrella-Balderrama, A., Fowler, J.J., Frati, F., Geyer, M., Gutwenger, C., Hong, S., Kaufmann, M., Kobourov, S.G., Liotta, G., Mutzel, P., Symvonis, A.: Colored simultaneous geometric embeddings and universal pointsets. *Algorithmica* **60**(3), 569–592 (2011)
9. Chambers, E.W., Eppstein, D., Goodrich, M.T., Löffler, M.: Drawing graphs in the plane with a prescribed outer face and polynomial area. *J. Graph Algorithms Appl.* **16**(2), 243–259 (2012)
10. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, Englewood Cliffs (1999)
11. Di Giacomo, E., Didimo, W., Liotta, G., Meijer, H., Trotta, F., Wismath, S.K.: k-colored point-set embeddability of outerplanar graphs. *J. Graph Algorithms Appl.* **12**(1), 29–49 (2008)
12. Di Giacomo, E., Liotta, G., Trotta, F.: Drawing colored graphs with constrained vertex positions and few bends per edge. *Algorithmica* **57**(4), 796–818 (2010)
13. Duncan, C.A., Goodrich, M.T., Kobourov, S.G.: Planar drawings of higher-genus graphs. *J. Graph Algorithms Appl.* **15**(1), 7–32 (2011)
14. Fößmeier, U., Kaufmann, M.: Nice drawings for planar bipartite graphs. In: Bongiovanni, G., Bovet, D.P., Di Battista, G. (eds.) CIAC 1997. LNCS, vol. 1203, pp. 122–134. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-62592-5_66
15. Goac, X., Kratochvíl, J., Okamoto, Y., Shin, C., Spillner, A., Wolff, A.: Untangling a planar graph. *Discrete Comput. Geom.* **42**(4), 542–569 (2009)
16. Halperin, D.: Arrangements. In: Goodman, J.E., O’Rourke, J. (eds.) *Handbook of Discrete and Computational Geometry*, Chap. 24, pp. 529–562. CRC Press LLC, Boca Raton (2004)

17. Harary, F.: Graph Theory. Addison-Wesley, Reading (1972)
18. Hong, S.H., Nagamochi, H.: Convex drawings of graphs with non-convex boundary constraints. *Discrete Appl. Math.* **156**(12), 2368–2380 (2008)
19. Ito, T., Misue, K., Tanaka, J.: Sphere anchored map: a visualization technique for bipartite graphs in 3D. In: Jacko, J.A. (ed.) HCI 2009. LNCS, vol. 5611, pp. 811–820. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02577-8_89
20. Kaufmann, M., Wagner, D. (eds.): Drawing Graphs. LNCS, vol. 2025. Springer, Heidelberg (2001). <https://doi.org/10.1007/3-540-44969-8>
21. Kaufmann, M., Wiese, R.: Embedding vertices at points: few bends suffice for planar graphs. *J. Graph Algorithms Appl.* **6**(1), 115–129 (2002)
22. Kindermann, P., Niedermann, B., Rutter, I., Schaefer, M., Schulz, A., Wolff, A.: Multi-sided boundary labeling. *Algorithmica* **76**(1), 225–258 (2016)
23. Lin, C.: Crossing-free many-to-one boundary labeling with hyperleaders. In: IEEE Pacific Visualization Symposium PacificVis 2010, Taipei, Taiwan, 2–5 March 2010, pp. 185–192. IEEE Computer Society (2010)
24. Mchedlidze, T., Nöllenburg, M., Rutter, I.: Extending convex partial drawings of graphs. *Algorithmica* **76**(1), 47–67 (2016)
25. McKee, T.A., McMorris, F.R.: Topics in Intersection Graph Theory. SIAM Monographs on Discrete Mathematics and Applications (1999)
26. Misue, K.: Anchored map: Graph drawing technique to support network mining. *IEICE Trans.* **91-D**(11), 2599–2606 (2008)
27. Misue, K., Zhou, Q.: Drawing semi-bipartite graphs in anchor+matrix style. In: Information Visualisation (IV 2011), London, UK, 13–15 July 2011, pp. 26–31. IEEE Computer Society (2011)
28. Neyer, G.: Map labeling with application to graph drawing. In: Kaufmann, M., Wagner, D. (eds.) Drawing Graphs. LNCS, vol. 2025, pp. 247–273. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44969-8_10
29. Pach, J., Wenger, R.: Embedding planar graphs at fixed vertex locations. *Graphs and Combinatorics* **17**(4), 717–728 (2001)
30. Patrignani, M.: On extending a partial straight-line drawing. *Int. J. Found. Comput. Sci.* **17**(5), 1061–1070 (2006)
31. Patrignani, M.: On extending a partial straight-line drawing. In: Healy, P., Nikolov, N.S. (eds.) GD 2005. LNCS, vol. 3843, pp. 380–385. Springer, Heidelberg (2006). https://doi.org/10.1007/11618058_34
32. Purchase, H.C.: Metrics for graph drawing aesthetics. *J. Vis. Lang. Comput.* **13**(5), 501–516 (2002)
33. Purchase, H.C., Carrington, D.A., Allder, J.: Empirical evaluation of aesthetics-based graph layout. *Empirical Softw. Eng.* **7**(3), 233–255 (2002)
34. Tamassia, R. (ed.): Handbook on Graph Drawing and Visualization. Chapman and Hall/CRC, Boca Raton (2013)
35. Tamassia, R., Liotta, G.: Graph drawing. In: Goodman, J.E., O’Rourke, J. (eds.) Handbook of Discrete and Computational Geometry, 2nd edn., pp. 1163–1185. Chapman and Hall/CRC, Boca Raton (2004)
36. Tutte, W.T.: How to draw a graph. *Proc. London Math. Soc.* **13**(3), 743–768 (1963)
37. Wolff, A., Strijk, T.: The map-labeling bibliography (1996). <http://i11www.ira.uka.de/map-labeling/bibliography>

Ordered Level Planarity, Geodesic Planarity and Bi-Monotonicity

Boris Klemz^(✉) and Günter Rote

Institute of Computer Science, Freie Universität Berlin, Berlin, Germany
klemz@inf.fu-berlin.de

Abstract. We introduce and study the problem ORDERED LEVEL PLANARITY which asks for a planar drawing of a graph such that vertices are placed at prescribed positions in the plane and such that every edge is realized as a y -monotone curve. This can be interpreted as a variant of LEVEL PLANARITY in which the vertices on each level appear in a prescribed total order. We establish a complexity dichotomy with respect to both the maximum degree and the level-width, that is, the maximum number of vertices that share a level. Our study of ORDERED LEVEL PLANARITY is motivated by connections to several other graph drawing problems.

GEODESIC PLANARITY asks for a planar drawing of a graph such that vertices are placed at prescribed positions in the plane and such that every edge e is realized as a polygonal path p composed of line segments with two adjacent directions from a given set S of directions symmetric with respect to the origin. Our results on ORDERED LEVEL PLANARITY imply \mathcal{NP} -hardness for any S with $|S| \geq 4$ even if the given graph is a matching. Katz, Krug, Rutter and Wolff claimed that for matchings MANHATTAN GEODESIC PLANARITY, the case where S contains precisely the horizontal and vertical directions, can be solved in polynomial time [GD 2009]. Our results imply that this is incorrect unless $\mathcal{P} = \mathcal{NP}$. Our reduction extends to settle the complexity of the BI-MONOTONICITY problem, which was proposed by Fulek, Pelsmajer, Schaefer and Štefankovič.

ORDERED LEVEL PLANARITY turns out to be a special case of T-LEVEL PLANARITY, CLUSTERED LEVEL PLANARITY and CONSTRAINED LEVEL PLANARITY. Thus, our results strengthen previous hardness results. In particular, our reduction to CLUSTERED LEVEL PLANARITY generates instances with only two non-trivial clusters. This answers a question posed by Angelini, Da Lozzo, Di Battista, Frati and Roselli.

Due to space constraints, some proofs in this manuscript are only sketched or omitted entirely. Full proofs of all claims can be found in the appendix of the preprint [17].

1 Introduction

In this paper we introduce ORDERED LEVEL PLANARITY and study its complexity. We establish connections to several other graph drawing problems, which we survey in this first section. We proceed from general problems to more and more constrained ones.

Upward Planarity: An *upward* planar drawing of a directed graph is a plane drawing where every edge $e = (u, v)$ is realized as a y -monotone curve that goes upward from u to v . Such drawings provide a natural way of visualizing a partial order on a set of items. The problem UPWARD PLANARITY of testing whether a directed graph has an upward planar drawing is \mathcal{NP} -complete [10]. However, if the y -coordinate of each vertex is prescribed, the problem can be solved in polynomial time [15]. This is captured by the notion of level graphs.

Level Planarity: A *level graph* $\mathcal{G} = (G, \gamma)$ is a directed graph $G = (V, E)$ together with a *level assignment* $\gamma : V \rightarrow \{0, \dots, h\}$ where γ is a surjective map with $\gamma(u) < \gamma(v)$ for every edge $(u, v) \in E$. Value h is the *height* of \mathcal{G} . The vertex set $V_i = \{v \mid \gamma(v) = i\}$ is called the i -th *level* of \mathcal{G} and $\lambda_i = |V_i|$ is its *width*. The *level-width* λ of \mathcal{G} is the maximum width of any level in \mathcal{G} . A *level planar drawing* of \mathcal{G} is an upward planar drawing of G where the y -coordinate of each vertex v is $\gamma(v)$. The horizontal line with y -coordinate i is denoted by L_i . The problem LEVEL PLANARITY asks whether a given level graph has a level planar drawing. The study of the complexity of LEVEL PLANARITY has a long history [7, 9, 13–15], culminating in a linear-time approach [15]. LEVEL PLANARITY has been extended to drawings of level graphs on surfaces different from the plane such as standing cylinder, a rolling cylinder or a torus [1, 3, 4].

An important special case are *proper* level graphs, that is, level graphs in which $\gamma(v) = \gamma(u) + 1$ for every edge $(u, v) \in E$. Instances of LEVEL PLANARITY can be assumed to be proper without loss of generality by subdividing long edges [7, 15]. However, in variations of LEVEL PLANARITY where we impose additional constraints, the assumption that instances are proper can have a strong impact on the complexity of the respective problems [2].

Level Planarity with Various Constraints: CLUSTERED LEVEL PLANARITY is a combination of CLUSTER PLANARITY and LEVEL PLANARITY. The task is to find a level planar drawing while simultaneously visualizing a given cluster hierarchy according to the rules of CLUSTER PLANARITY. The problem is \mathcal{NP} -complete in general [2], but efficiently solvable for proper instances [2, 8].

T-LEVEL PLANARITY is a consecutivity-constrained version of LEVEL PLANARITY: every level V_i is equipped with a tree T_i whose set of leaves is V_i . For every inner node u of T_i the leaves of the subtree rooted at u have to appear consecutively along L_i . The problem is \mathcal{NP} -complete in general [2], but efficiently solvable for proper instances [2, 18]. The precise definitions of both problems and a longer discussion about the related work can be found in [17].

Very recently, Brückner and Rutter [6] explored a variant of LEVEL PLANARITY in which the left-to-right order of the vertices on each level has to

be a linear extension of a given partial order. They refer to this problem as **CONSTRAINED LEVEL PLANARITY** and they provide an efficient algorithm for single-source graphs and show \mathcal{NP} -completeness of the general case.

A Common Special Case - Ordered Level Planarity: We introduce a natural variant of **LEVEL PLANARITY** that specifies a total order for the vertices on each level. An *ordered level graph* \mathcal{G} is a triple $(G = (V, E), \gamma, \chi)$ where (G, γ) is a level graph and $\chi : V \rightarrow \{0, \dots, \lambda - 1\}$ is a *level ordering* for G . We require that χ restricted to domain V_i bijectively maps to $\{0, \dots, \lambda_i - 1\}$. An *ordered level planar drawing* of an ordered level graph \mathcal{G} is a level planar drawing of (G, γ) where for every $v \in V$ the x -coordinate of v is $\chi(v)$. Thus, the position of every vertex is fixed. The problem **ORDERED LEVEL PLANARITY** asks whether a given ordered level graph has an ordered level planar drawing.

In the above definitions, the x - and y -coordinates assigned via χ and γ merely act as a convenient way to encode total and partial orders respectively. In terms of realizability, the problems are equivalent to generalized versions where χ and γ map to the reals. In other words, the fixed vertex positions can be any points in the plane. All reductions and algorithms in this paper carry over to these generalized versions, if we pay the cost for presorting the vertices according to their coordinates. **ORDERED LEVEL PLANARITY** is also equivalent to a relaxed version where we only require that the vertices of each level V_i appear along L_i according to the given total order without insisting on specific coordinates. We make use of this equivalence in many of our figures for the sake of visual clarity.

Geodesic Planarity: Let $S \subset \mathbb{Q}^2$ be a finite set of directions symmetric with respect to the origin, i.e. for each direction $s \in S$, the reverse direction $-s$ is also contained in S . A plane drawing of a graph is *geodesic* with respect to S if every edge is realized as a polygonal path p composed of line segments with two adjacent directions from S . Two directions of S are *adjacent* if they appear consecutively in the projection of S to the unit circle. Such a path p is a geodesic with respect to some polygonal norm that corresponds to S . An instance of the decision problem **GEODESIC PLANARITY** is a 4-tuple $\mathcal{G} = (G = (V, E), x, y, S)$ where G is a graph, x and y map from V to the reals and S is a set of directions as stated above. The task is to decide whether \mathcal{G} has a *geodesic drawing*, that is, G has a geodesic drawing with respect to S in which every vertex $v \in V$ is placed at $(x(v), y(v))$.

Katz et al. [16] study **MANHATTAN GEODESIC PLANARITY**, which is the special case of **GEODESIC PLANARITY** where the set S consists of the two horizontal and the two vertical directions. Geodesic drawings with respect to this set of direction are also referred to as orthogeodesic drawings [11, 12]. Katz et al. [16] show that a variant of **MANHATTAN GEODESIC PLANARITY** in which the drawings are restricted to the integer grid is \mathcal{NP} -hard even if G is a perfect matching. The proof is by reduction from **3-PARTITION** and makes use of the fact the number of edges that can pass between two vertices on a grid line is bounded. In contrast, they claim that the standard version of **MANHATTAN GEODESIC PLANARITY** is polynomial-time solvable for perfect matchings [16, Theorem 5]. To this end, they sketch a plane sweep algorithm that maintains

a linear order among the edges that cross the sweep line. When a new edge is encountered it is inserted as low as possible subject to the constraints implied by the prescribed vertex positions. When we asked the authors for more details, they informed us that they are no longer convinced of the correctness of their approach. Theorem 2 of our paper implies that the approach is indeed incorrect unless $\mathcal{P} = \mathcal{NP}$.

Bi-Monotonicity: Fulek et al. [9] present a Hanani-Tutte theorem for *y-monotone* drawings, that is, upward drawings in which all vertices have distinct *y*-coordinates. They accompany their result with a simple and efficient algorithm for Y-MONOTONICITY, which is equivalent to LEVEL PLANARITY restricted to instances with level-width $\lambda = 1$. They propose the problem BI-MONOTONICITY and leave its complexity as an open problem. The input of BI-MONOTONICITY is a triple $\mathcal{G} = (G = (V, E), x, y)$ where G is a graph and x and y injectively map from V to the reals. The task is to decide whether \mathcal{G} has a bi-monotone drawing, that is, a plane drawing in which edges are realized as curves that are both *y*-monotone and *x*-monotone and in which every vertex $v \in V$ is placed at $(x(v), y(v))$.

Main Results: In Sect. 3 we study the complexity of ORDERED LEVEL PLANARITY. While UPWARD PLANARITY is \mathcal{NP} -complete [10] in general but becomes polynomial-time solvable [15] for prescribed *y*-coordinates, we show that prescribing both *x*-coordinates and *y*-coordinates renders the problem \mathcal{NP} -complete. We complement our result with efficient approaches for some special cases of ordered level graphs and, thereby, establish a complexity dichotomy with respect to the level-width and the maximum degree.

Theorem 1. ORDERED LEVEL PLANARITY is \mathcal{NP} -complete, even for maximum degree $\Delta = 2$ and level-width $\lambda = 2$. For level-width $\lambda = 1$ or $\Delta^+ = \Delta^- = 1$ or proper instances ORDERED LEVEL PLANARITY can be solved in linear time, where Δ^+ and Δ^- are the maximum in-degree and out-degree respectively.

ORDERED LEVEL PLANARITY restricted to instances with $\lambda = 2$ and $\Delta = 2$ is an elementary problem. We expect that it may serve as a suitable basis for future reductions. As a proof of concept, the remainder of this paper is devoted to establishing connections between ORDERED LEVEL PLANARITY and several other graph drawing problems. Theorem 1 serves as our key tool for settling their complexity. In Sect. 2 we study GEODESIC PLANARITY and obtain:

Theorem 2. GEODESIC PLANARITY is \mathcal{NP} -hard for any set of directions S with $|S| \geq 4$ even for perfect matchings in general position.

Observe the aforementioned discrepancy between Theorem 2 and the claim by Katz et al. [16] that MANHATTAN GEODESIC PLANARITY for perfect matchings is in \mathcal{P} . BI-MONOTONICITY is closely related to a special case of MANHATTAN GEODESIC PLANARITY. With a simple corollary we settle the complexity of BI-MONOTONICITY and, thus, answer the open question by Fulek et al. [9].

Theorem 3. BI-MONOTONICITY is \mathcal{NP} -hard even for perfect matchings.

ORDERED LEVEL PLANARITY is an immediate and very constrained special case of CONSTRAINED PLANARITY. Further, we establish ORDERED LEVEL PLANARITY as a special case of both CLUSTERED LEVEL PLANARITY and T-LEVEL PLANARITY by providing the following reductions.

Theorem 4. ORDERED LEVEL PLANARITY with maximum degree $\Delta = 2$ and level-width $\lambda = 2$ reduces in linear time to T-LEVEL PLANARITY with maximum degree $\Delta' = 2$ and level-width $\lambda' = 4$.

Theorem 5. ORDERED LEVEL PLANARITY with maximum degree $\Delta = 2$ and level-width $\lambda = 2$ reduces in quadratic time to CLUSTERED LEVEL PLANARITY with maximum degree $\Delta' = 2$, level-width $\lambda' = 2$ and $\kappa' = 3$ clusters.

Angelini et al. [2] propose the complexity of CLUSTERED LEVEL PLANARITY for clustered level graphs with a flat cluster hierarchy as an open question. Theorem 5 answers this question by showing that \mathcal{NP} -hardness holds for instances with only two non-trivial clusters.

2 Geodesic Planarity and Bi-Monotonicity

In this section we establish that deciding whether an instance $\mathcal{G} = (G, x, y, S)$ of GEODESIC PLANARITY has a geodesic drawing is \mathcal{NP} -hard even if G is a perfect matching and even if the coordinates assigned via x and y are in *general position*, that is, no two vertices lie on a line with a direction from S . The \mathcal{NP} -hardness of BI-MONOTONICITY for perfect matchings follows as a simple corollary. Our results are obtained via a reduction from ORDERED LEVEL PLANARITY.

Lemma 1. Let $S \subset \mathbb{Q}^2$ with $|S| \geq 4$ be a finite set of directions symmetric with respect to the origin. ORDERED LEVEL PLANARITY with maximum degree $\Delta = 2$ and level-width $\lambda = 2$ reduces to GEODESIC PLANARITY such that the resulting instances are in general position and consist of a perfect matching and direction set S . The reduction can be carried out using a linear number of arithmetic operations.

Proof Sketch. In this sketch, we prove our claim only for the classical case that S contains exactly the four horizontal and vertical directions. Our reduction is carried out in two steps. Let $\mathcal{G}_o = (G_o = (V, E), \gamma, \chi)$ be an ORDERED LEVEL PLANARITY instance with maximum degree $\Delta = 2$ and level-width $\lambda = 2$. In Step (i) we turn \mathcal{G}_o into an equivalent GEODESIC PLANARITY instance $\mathcal{G}'_g = (G_o, x', \gamma, S)$. In Step (ii) we transform \mathcal{G}'_g into an equivalent GEODESIC PLANARITY instance $\mathcal{G}_g = (G_g, x, y, S)$ where G_g is a perfect matching and the vertex positions assigned via x and y are in general position.

Step (i): In order to transform \mathcal{G}_o into \mathcal{G}'_g we apply a shearing transformation. We translate the vertices of each level V_i by $3i$ units to the right, see Fig. 1(a)

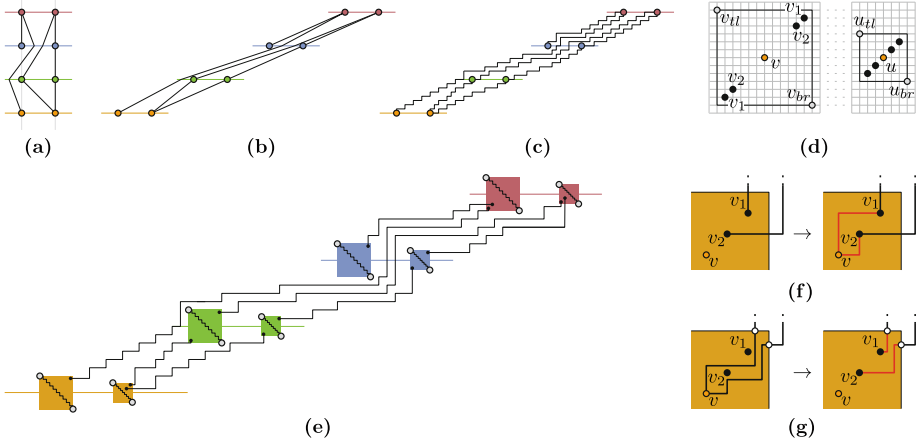


Fig. 1. (a), (b) and (c): Illustrations of Step (i). (d) The two gadget squares of each level. Grid cells have size $1/48 \times 1/48$. (e) Illustration of Step (ii). Turning a drawing of \mathcal{G}_g into a drawing of \mathcal{G}'_g (f) and vice versa (g).

and (b). Clearly, every geodesic drawing of \mathcal{G}'_g can be turned into an ordered level planar drawing of \mathcal{G}_o . On the other hand, consider an ordered level planar drawing Γ_o of \mathcal{G}_o . Without loss of generality we can assume that in Γ_o all edges are realized as polygonal paths in which bend points occur only on the horizontal lines L_i through the levels V_i where $0 \leq i \leq h$. Further, we may assume that all bend points have x -coordinates in the open interval $(-1, 2)$. We shear Γ_o by translating the bend points and vertices of level V_i by $3i$ units to the right for $0 \leq i \leq h$, see Fig. 1(b). In the resulting drawing Γ'_o , the vertex positions match those of \mathcal{G}'_g . Furthermore, all edge-segments have a positive slope. Thus, since the maximum degree is $\Delta = 2$ we can replace all edge-segments with L_1 -geodesic rectilinear paths that closely trace the segments and we obtain a geodesic drawing Γ'_g of \mathcal{G}'_g , see Fig. 1(c).

Step (ii): In order to turn $\mathcal{G}'_g = (G_o = (V, E), x', \gamma, S)$ into the equivalent instance $\mathcal{G}_g = (G_g, x, y, S)$ we transform G_o into a perfect matching. To this end, we split each vertex $v \in V$ by replacing it with a small gadget that fits inside a square r_v centered on the position $p_v = (x'(v), \gamma(v))$ of v , see Fig. 1(e). We call r_v the *square* of v and use $p_v^{tr}, p_v^{tl}, p_v^{br}$ and p_v^{bl} to denote the top-right, top-left, bottom-right and bottom-left corner of r_v , respectively. We use two different sizes to ensure general position. The size of the gadget square is $1/4 \times 1/4$ if $\chi(v) = 0$ and it is $1/8 \times 1/8$ if $\chi(v) = 1$. The gadget contains a degree-1 vertex for every edge incident to v . In the following we explain the gadget construction in detail, for an illustration see Fig. 1(d). Let $\{v, u\}$ be an edge incident to v . We create an edge $\{v_1, u\}$ where v_1 is a new vertex which is placed at $p_v^{tr} - (1/48, 1/48)$ if u is located to the top-right of v and it is placed at $p_v^{bl} + (1/48, 1/48)$ if u is located to the bottom-left of v . Similarly,

if v is incident to a second edge $\{v, u'\}$, we create an edge $\{v_2, u'\}$ where v_2 is placed at $p_v^{tr} - (1/24, 1/24)$ or $p_v^{bl} + (1/24, 1/24)$ depending on the position of u' . Finally, we create a *blocking* edge $\{v_{tl}, v_{br}\}$ where v_{tl} is placed at p_v^{tl} and v_{br} is placed at p_v^{br} . The thereby assigned coordinates are in general position and the construction can be carried out in linear time.

Assume that \mathcal{G}_g has a geodesic drawing Γ_g . By construction, all blocking edges have a top-left and a bottom-right endpoint. On the other hand, all other edges have a bottom-left and a top-right endpoint. As a result, a non-blocking edge $e = \{u, v\}$ can not pass through any gadget square r_w , except the squares r_u or r_v since e would have to cross the blocking edge of r_w . Accordingly, it is straight-forward to obtain a geodesic drawing of Γ'_g : We remove the blocking edges, reinsert the vertices of V according to the mappings x' and γ and connect them to the vertices of their respective gadgets in a geodesic fashion. This can always be done without crossings. Figure 1(f) shows one possibility. If the edge from v_2 passes to the left of v_1 , we may have to choose a reflected version. Finally, we remove the vertices v_1 and v_2 which now act as subdivision vertices.

On the other hand, let Γ'_g be a geodesic planar drawing of \mathcal{G}'_g . Without loss of generality, we can assume that each edge $\{u, v\}$ passes only through the squares of u and v . Furthermore, for each $v \in V$ we can assume that its incident edges intersect the boundary of r_v only to the top-right of $p_v^{tr} - (1/48, 1/48)$ or to the bottom-left of $p_v^{bl} + (1/48, 1/48)$, see Fig. 1(g). Thus, we can simply remove the parts of the edges in the interior of the gadget squares and connect the gadget vertices to the intersection points of the edges with the gadget squares in a geodesic fashion. \square

The bit size of the numbers involved in the calculations of our reduction is linearly bounded in the bit size of the directions of S . Together with Theorem 1 we obtain the proof of Theorem 2. The instances generated by Lemma 1 are in general position. In particular, this means that the mappings x and y are injective. We obtain an immediate reduction to BI-MONOTONICITY. The correctness follows from the fact that every L_1 -geodesic rectilinear path can be transformed into a bi-monotone curve and vice versa. Thus, we obtain Theorem 3.

3 Ordered Level Planarity

To show \mathcal{NP} -hardness of ORDERED LEVEL PLANARITY we reduce from a 3-SATISFIABILITY variant described in this paragraph. A *monotone* 3-SATISFIABILITY formula is a Boolean 3-SATISFIABILITY formula in which each clause is either *positive* or *negative*, that is, each clause contains either exclusively positive or exclusively negative literals respectively. A *planar* 3SAT formula $\varphi = (\mathcal{U}, \mathcal{C})$ is a Boolean 3-SATISFIABILITY formula with a set \mathcal{U} of variables and a set \mathcal{C} of clauses such that its *variable-clause graph* $G_\varphi = (\mathcal{U} \uplus \mathcal{C}, E)$ is planar. The graph G_φ is bipartite, i.e. every edge in E is incident to both a *clause* vertex from \mathcal{C} and a *variable* vertex from \mathcal{U} . Furthermore, edge $\{c, u\} \in E$ if and only if a literal of variable $u \in \mathcal{U}$ occurs in $c \in \mathcal{C}$. PLANAR MONOTONE

3-SATISFIABILITY is a special case of 3-SATISFIABILITY where we are given a planar and monotone 3-SATISFIABILITY formula φ and a *monotone rectilinear representation* \mathcal{R} of the variable-clause graph of φ . The representation \mathcal{R} is a contact representation on an integer grid in which the variables are represented by horizontal line segments arranged on a line ℓ . The clauses are represented by E-shapes turned by 90° such that all positive clauses are placed above ℓ and all negative clauses are placed below ℓ , see Fig. 2a. PLANAR MONOTONE 3-SATISFIABILITY is \mathcal{NP} -complete [5]. We are now equipped to prove the core lemma of this section.

Lemma 2. PLANAR MONOTONE 3-SATISFIABILITY *reduces in polynomial time to ORDERED LEVEL PLANARITY. The resulting instances have maximum degree $\Delta = 2$ and all vertices on levels with width at least 3 have out-degree at most 1 and in-degree at most 1.*

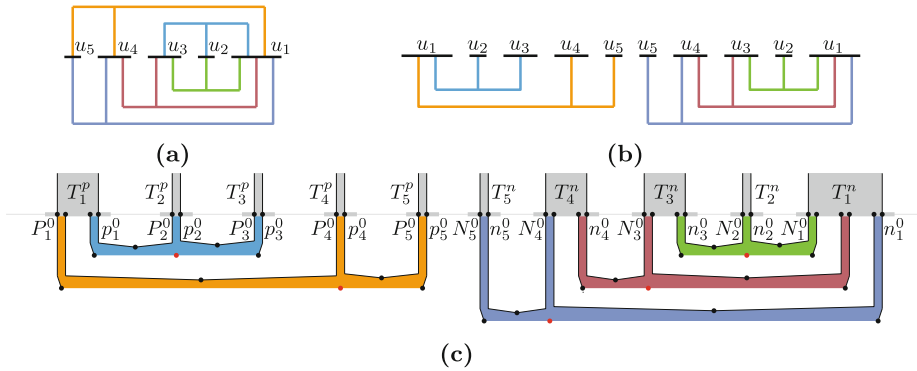


Fig. 2. (a) Representation \mathcal{R} of φ with negative clauses $(\bar{u}_1 \vee \bar{u}_4 \vee \bar{u}_5)$, $(\bar{u}_1 \vee \bar{u}_3 \vee \bar{u}_4)$ and $(\bar{u}_1 \vee \bar{u}_2 \vee \bar{u}_3)$ and positive clauses $(u_1 \vee u_4 \vee u_5)$ and $(u_1 \vee u_2 \vee u_3)$ and (b) its modified version \mathcal{R}' in Lemma 2. (c) Tier \mathcal{T}_0 .

Proof Sketch. We perform a polynomial-time reduction from PLANAR MONOTONE 3-SATISFIABILITY. Let $\varphi = (\mathcal{U}, \mathcal{C})$ be a planar and monotone 3-SATISFIABILITY formula with $\mathcal{C} = \{c_1, \dots, c_{|\mathcal{C}|}\}$. Let G_φ the variable-clause graph of φ . Let \mathcal{R} be a monotone rectilinear representation of G_φ . We construct an ordered level graph $\mathcal{G} = (G, \gamma, \chi)$ such that \mathcal{G} has an ordered level planar drawing if and only if φ is satisfiable. In this proof sketch we omit some technical details such as precise level assignments and level orderings.

Overview: The ordered level graph \mathcal{G} has $l_3 + 1$ levels which are partitioned into four tiers $\mathcal{T}_0 = \{0, \dots, l_0\}$, $\mathcal{T}_1 = \{l_0 + 1, \dots, l_1\}$, $\mathcal{T}_2 = \{l_1 + 1, \dots, l_2\}$ and $\mathcal{T}_3 = \{l_2 + 1, \dots, l_3\}$. Each clause $c_i \in \mathcal{C}$ is associated with a clause edge $e_i = (c_i^s, c_i^t)$ starting with c_i^s in tier \mathcal{T}_0 and ending with c_i^t in tier \mathcal{T}_2 . The clause edges have to be drawn in a system of tunnels that encodes the 3-SATISFIABILITY formula φ . In \mathcal{T}_0 the layout of the tunnels corresponds directly

to the rectilinear representation \mathcal{R} , see Fig. 2c. For each E-shape there are three tunnels corresponding to the three literals of the associated clause. The bottom vertex c_i^s of each clause edge e_i is placed such that e_i has to be drawn inside one of the three tunnels of the E-shape corresponding to c_i . This corresponds to the fact that in a satisfying truth assignment every clause has at least one satisfied literal. In tier \mathcal{T}_1 we merge all the tunnels corresponding to the same literal. We create variable gadgets that ensure that for each variable u edges of clauses containing u can be drawn in the tunnel associated with either the negative or the positive literal of u but not both. This corresponds to the fact that every variable is set to either true or false. Tiers \mathcal{T}_2 and \mathcal{T}_3 have a technical purpose.

We proceed by describing the different tiers in detail. Recall that in terms of realizability, ORDERED LEVEL PLANARITY is equivalent to the generalized version where γ and χ map to the reals. For the sake of convenience we will begin by designing \mathcal{G} in this generalized setting. It is easy to transform \mathcal{G} such that it satisfies the standard definition in a polynomial-time post processing step.

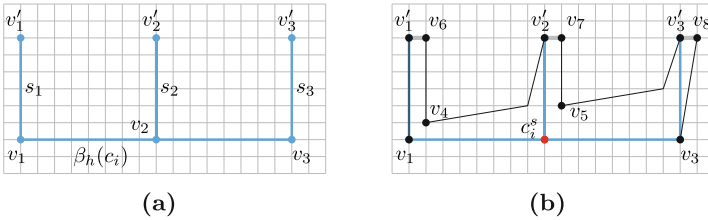


Fig. 3. (a) The E-shape and (b) the clause gadget of clause c_i . The thick gray lines represent the gates of c_i .

Tier 0 and 2, Clause Gadgets: The clause edges $e_i = (c_i^s, c_i^t)$ end in tier \mathcal{T}_2 . It is composed of $l_2 - l_1 = |\mathcal{C}|$ levels each of which contains precisely one vertex. We assign $\gamma(c_i^t) = l_1 + i$. Observe that this imposes no constraint on the order in which the edges enter \mathcal{T}_2 .

Tier \mathcal{T}_0 consists of a system of tunnels that resembles the monotone rectilinear representation \mathcal{R} of $G_\varphi = (\mathcal{U} \uplus \mathcal{C}, E)$, see Fig. 2c. Intuitively it is constructed as follows: We take the top part of \mathcal{R} , rotate it by 180° and place it to the left of the bottom part such that the variables' line segments align, see Fig. 2b. We call the resulting representation \mathcal{R}' . For each E-shape in \mathcal{R}' we create a *clause gadget*, which is a subgraph composed of 11 vertices that are placed on a grid close to the E-shape, see Fig. 3. The red vertex at the bottom is the lower vertex c_i^s of the clause edge e_i of the clause c_i corresponding to the E-shape. Without loss of generality we assume the grid to be fine enough such that the resulting ordered level graph can be drawn as in Fig. 2c without crossings. Further, we assume that the y -coordinates of every pair of horizontal segments belonging to distinct E-shapes differ by at least 3. This ensures that all vertices on levels with width at least 3 have out-degree at most 1 and in-degree at most 1 as stated in the lemma.

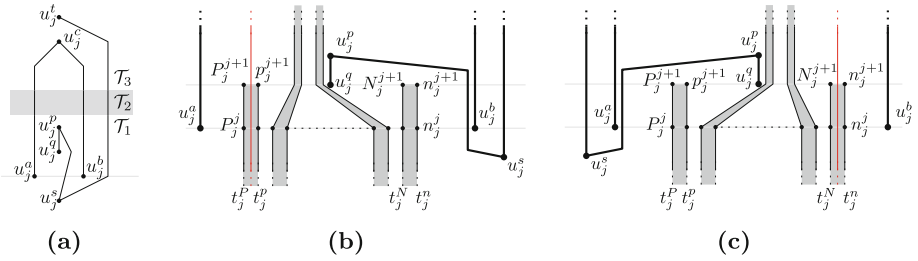


Fig. 4. (a) The variable gadget of u_j in (b) positive and (c) negative state.

The clause gadget (without the clause edge) has a *unique* ordered level planar drawing in the sense that for every level V_i the left-to-right sequence of vertices and edges intersected by the horizontal line L_i through V_i is identical in every ordered level planar drawing. This is due to the fact that the order of the top-most vertices $v_1', v_6, v_2', v_7, v_3'$ and v_8 is fixed. We call the line segments $v_1'v_6, v_2'v_7$ and $v_3'v_8$ the *gates* of c_i . Note that the clause edge e_i has to intersect one of the gates of c_i . This corresponds to the fact that at least one literal of every clause has to be satisfied.

The subgraph G_0 induced by \mathcal{T}_0 (without the clause edges) has a unique ordered level planar drawing. In tier \mathcal{T}_1 we bundle all gates that belong to one literal together by creating two long paths for each literal. These two paths form the *tunnel* of the corresponding literal. All clause edges intersecting a gate of some literal have to be drawn inside the literal's tunnel, see Fig. 2c. To this end, for $j = 1, \dots, |\mathcal{U}|$ we use N_j^0 (n_j^0) to refer to the left-most (right-most) vertex of a negative clause gadget placed on a line segment of \mathcal{R}' representing $u_j \in \mathcal{U}$. The vertices N_j^0 and n_j^0 are the first vertices of the paths forming the *negative* tunnel T_j^n of the negative literal of variable u_j . Analogously, we use P_j^0 (p_j^0) to refer to the left-most (right-most) vertex of a positive clause gadget placed on a line segment of \mathcal{R}' representing u_j . The vertices P_j^0 and p_j^0 are the first vertices of the paths forming the *positive* tunnel T_j^p of the positive literal of variable u_j . If for some j the variable u_j is not contained both in negative and positive clauses, we artificially add two vertices N_j^0 and n_j^0 or P_j^0 and p_j^0 on the corresponding line segments in order to avoid having to treat special cases in the remainder of the construction.

Tier 1 and 3, Variable Gadgets: Recall that every clause edge has to pass through a gate that is associated with some literal of the clause, and, thus, every edge is drawn in the tunnel of some literal. We need to ensure that it is not possible to use tunnels associated with the positive, as well as the negative literal of some variable simultaneously. To this end, we create a *variable gadget* with vertices in tier \mathcal{T}_1 and tier \mathcal{T}_3 for each variable. The variable gadget of variable u_j is illustrated in Fig. 4a. The variable gadgets are nested in the sense that they start in \mathcal{T}_1 in the order $u_1, u_2, \dots, u_{|\mathcal{U}|}$, from bottom to top and they end in the reverse order in \mathcal{T}_3 , see Fig. 5. We force all tunnels with index at least j to

be drawn between the vertices u_j^a and u_j^b . This is done by subdividing the tunnel edges on this level, see Fig. 4b. The *long edge* (u_j^s, u_j^t) has to be drawn to the left or right of u_j^c in \mathcal{T}_3 . Accordingly, it is drawn to the left of u_j^a or to the right of u_j^b in \mathcal{T}_1 . Thus, it is drawn either to the right (Fig. 4b) of all the tunnels or to the left (Fig. 4c) of all the tunnels. As a consequence, the *blocking edge* (u_j^s, u_j^t) is also drawn either to the right or the left of all the tunnels. Together with the edge (u_j^q, u_j^p) it prevents clause edges from being drawn either in the positive tunnel T_j^p or negative tunnel T_j^n of variable u_j which end at level $\gamma(u_j^q)$ because they can not reach their endpoints in \mathcal{T}_2 without crossings. We say T_j^p or T_j^n are *blocked* respectively.

The construction of the ordered level graph \mathcal{G} can be carried out in polynomial time. Note that maximum degree is $\Delta = 2$ and that all vertices on levels with width at least 3 have out-degree at most 1 and in-degree at most 1 as claimed in the lemma.

Correctness: It remains to show that \mathcal{G} has an ordered level planar drawing if and only if φ is satisfiable. Assume that \mathcal{G} has an ordered level planar drawing Γ . We create a satisfying truth assignment for φ . If T_j^n is blocked we set u_j to true, otherwise we set u_j to false for $j \in 1, \dots, |\mathcal{U}|$. Recall that the subgraph G_0 induced by the vertices in tier \mathcal{T}_0 has a unique ordered level planar drawing. Consider a clause c_i and let f, g, j be the indices of the variables whose literals are contained in c_i . Clause edge $e_i = (e_i^s, e_i^t)$ has to pass level l_0 through one of the gates of c_i . More precisely, it has to be drawn between either N_f^0 and n_g^0 ,

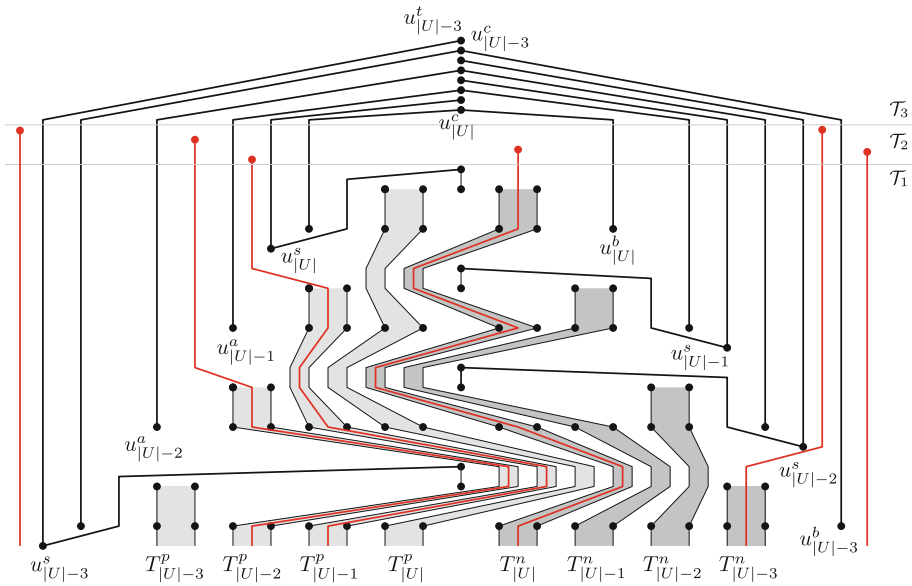


Fig. 5. Nesting structure of the variable gadgets.

N_g^0 and n_g^0 or N_j^0 and n_j^0 if c_i is negative or between either P_f^0 and p_f^0 , P_g^0 and p_g^0 or P_j^0 and p_j^0 if c_i is positive, see Fig. 2c. First, assume that c_i is negative and assume without loss of generality that it traverses l_0 between N_j^0 and n_j^0 . In this case clause edge e_i has to be drawn in T_j^n . Recall that this is only possible if T_j^n is not blocked, which is the case if u_j is false, see Fig. 4c. Analogously, if c_i is positive and e_i traverses w.l.o.g. between p_j^p and p_j^p , then u_j is true, Fig. 4b. Thus, we have established that one literal of each clause in \mathcal{C} evaluates to true for our truth assignment and, hence, formula φ is satisfiable.

Now assume that φ is satisfiable and consider a satisfying truth assignment. We create an ordered level planar drawing Γ of \mathcal{G} . It is clear how to create the unique subdrawing of G_0 . The variable gadgets are drawn in a nested fashion, see Fig. 5. For $j = 1, \dots, |\mathcal{U}| - 1$ we draw edge (u_j^a, u_j^c) to left of u_{j+1}^a and u_{j+1}^s and edge (u_j^b, u_j^e) to right of u_{j+1}^b and u_{j+1}^s . In other words, the pair $((u_j^a, u_j^c), (u_j^b, u_j^e))$ is drawn between all such pairs with index smaller than j . Recall that the vertices $u_j^a, u_j^b, u_j^s, u_j^t$ and u_j^q are located on higher levels than the according vertices of variables with index smaller than j and that u_j^t and u_j^c are located on lower levels than the according vertices of variables with index smaller than j .

For $j = 1, \dots, |\mathcal{U}|$ if u_j is positive we draw the long edge (u_j^s, u_j^t) to the right of u_j^b and u_j^c and, accordingly, we have to draw all tunnels left of u_j^s and u_j^q (except for T_j^n , which has to be drawn to the left of u_j^s and end to the right of u_j^q), see Fig. 4b. If u_j is negative we draw the long edge (u_j^s, u_j^t) to the left of u_j^b and u_j^c and, accordingly, we have to draw all tunnels right of u_j^s and u_j^q (except for T_j^p , which has to be drawn to the right of u_j^s and end to the left of u_j^q), see Fig. 4c. We have to draw the blocking edge (u_j^s, u_j^p) to the right of n_{j+1}^{j+1} if u_j is positive and to the left of P_{j+1}^{j+1} if u_j is negative.

It remains to describe how to draw the clause edges. Let c_i be a clause. There is at least one true literal in c_i . Let k be the index of the corresponding variable. We describe the drawing of clause edge $e_i = (c_i^s, c_i^t)$ from bottom to top. We start by drawing e_i in the tunnel T_k^p (T_k^n) if c_i is positive (negative). After the variable gadget of u_k the edge e_i leaves its tunnel and is drawn to the left (right) of all gadgets of variables with higher index, see Fig. 5. □

We obtain \mathcal{NP} -hardness for instances with maximum degree $\Delta = 2$. In fact, we can restrict our attention to instances level-width $\lambda = 2$. To this end, we split levels with width $\lambda_i > 2$ into $\lambda_i - 1$ levels containing exactly two vertices each.

Lemma 3. *An instance $\mathcal{G} = (G = (V, E), \gamma, \chi)$ of ORDERED LEVEL PLANARITY with maximum degree $\Delta \leq 2$ can be transformed in linear time into an equivalent instance $\mathcal{G}' = (G' = (V', E'), \gamma', \chi')$ of ORDERED LEVEL PLANARITY with level-width $\lambda' \leq 2$ and maximum degree Δ' . If in \mathcal{G} all vertices on levels with width at least 3 have out-degree at most 1 and in-degree at most 1, then $\Delta' \leq 2$. Otherwise, $\Delta' \leq \Delta + 1$.*

The reduction in Lemma 2 requires degree-2 vertices. With $\Delta = 1$, the problem becomes polynomial-time solvable. In fact, even if $\Delta = 2$ one can easily

solve it as long as the maximum in-degree and the maximum out-degree are both bounded by 1. Such instances consists of a set P of y -monotone paths.

We write $p \prec q$, meaning that $p \in P$ must be drawn to the left of $q \in P$, if p and q have vertices v_p and v_q that lie adjacent on a common level. If \prec is acyclic, we can draw \mathcal{G} according to a linear extension of \prec , otherwise there exists no solution.

Lemma 4. ORDERED LEVEL PLANARITY *restricted to instances with maximum in-degree $\Delta^- = 1$ and maximum out-degree $\Delta^+ = 1$ can be solved in linear time.*

For $\lambda = 1$ ORDERED LEVEL PLANARITY is solvable in linear time since LEVEL PLANARITY can be solved in linear time [15]. Proper instances can be solved in linear-time via a sweep through every level. The problem is obviously contained in \mathcal{NP} . The results of this section establish Theorem 1.

Acknowledgements. We thank the authors of [16] for providing us with unpublished information regarding their plane sweep approach for MANHATTAN GEODESIC PLANARITY.

References

1. Angelini, P., Da Lozzo, G., Di Battista, G., Frati, F., Patrignani, M., Rutter, I.: Beyond level planarity. In: Hu, Y., Nöllenburg, M. (eds.) GD 2016. LNCS, vol. 9801, pp. 482–495. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_37
2. Angelini, P., Da Lozzo, G., Di Battista, G., Frati, F., Roselli, V.: The importance of being proper: (in clustered-level planarity and T-level planarity). Theor. Comput. Sci. **571**, 1–9 (2015). <https://doi.org/10.1016/j.tcs.2014.12.019>
3. Bachmaier, C., Brandenburg, F., Forster, M.: Radial level planarity testing and embedding in linear time. J. Graph Algorithms Appl. **9**(1), 53–97 (2005). <http://jgaa.info/accepted/2005/BachmaierBrandenburgForster2005.9.1.pdf>
4. Bachmaier, C., Brunner, W.: Linear time planarity testing and embedding of strongly connected cyclic level graphs. In: Halperin, D., Mehlhorn, K. (eds.) ESA 2008. LNCS, vol. 5193, pp. 136–147. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87744-8_12
5. de Berg, M., Khosravi, A.: Optimal binary space partitions for segments in the plane. Int. J. Comput. Geometry Appl. **22**(3), 187–206 (2012). <http://www.worldscinet.com/doi/abs/10.1142/S0218195912500045>
6. Brückner, G., Rutter, I.: Partial and constrained level planarity. In: Klein, P.N. (ed.) Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, 16–19 January, pp. 2000–2011. SIAM (2017). <https://doi.org/10.1137/1.9781611974782>
7. Di Battista, G., Nardelli, E.: Hierarchies and planarity theory. IEEE Trans. Syst. Man Cybern. **18**(6), 1035–1046 (1988). <https://doi.org/10.1109/21.23105>
8. Forster, M., Bachmaier, C.: Clustered level planarity. In: Van Emde Boas, P., Pokorný, J., Bieliková, M., Štuller, J. (eds.) SOFSEM 2004. LNCS, vol. 2932, pp. 218–228. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24618-3_18

9. Fulek, R., Pelsmajer, M.J., Schaefer, M., Štefankovič, D.: Hanani-Tutte, monotone drawings, and level-planarity. In: Pach, J. (ed.) *Thirty Essays on Geometric Graph Theory*, pp. 263–287. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-0110-0_14
10. Garg, A., Tamassia, R.: On the computational complexity of upward and rectilinear planarity testing. *SIAM J. Comput.* **31**(2), 601–625 (2001). <https://doi.org/10.1137/S0097539794277123>
11. Giacomo, E.D., Frati, F., Fulek, R., Grilli, L., Krug, M.: Orthogeodesic point-set embedding of trees. *Comput. Geom.* **46**(8), 929–944 (2013). <https://doi.org/10.1016/j.comgeo.2013.04.003>
12. Giacomo, E.D., Grilli, L., Krug, M., Liotta, G., Rutter, I.: Hamiltonian orthogeodesic alternating paths. *J. Discrete Algorithms* **16**, 34–52 (2012). <https://doi.org/10.1016/j.jda.2012.04.012>
13. Heath, L.S., Pemmaraju, S.V.: Recognizing leveled-planar dags in linear time. In: Brandenburg, F.J. (ed.) *GD 1995*. LNCS, vol. 1027, pp. 300–311. Springer, Heidelberg (1996). <https://doi.org/10.1007/BFb0021813>
14. Jünger, M., Leipert, S., Mutzel, P.: Pitfalls of using PQ-trees in automatic graph drawing. In: DiBattista, G. (ed.) *GD 1997*. LNCS, vol. 1353, pp. 193–204. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63938-1_62
15. Jünger, M., Leipert, S., Mutzel, P.: Level planarity testing in linear time. In: Whitesides, S.H. (ed.) *GD 1998*. LNCS, vol. 1547, pp. 224–237. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-37623-2_17
16. Katz, B., Krug, M., Rutter, I., Wolff, A.: Manhattan-geodesic embedding of planar graphs. In: Eppstein, D., Gansner, E.R. (eds.) *GD 2009*. LNCS, vol. 5849, pp. 207–218. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11805-0_21
17. Klemz, B., Rote, G.: Ordered level planarity, geodesic planarity and Bi-Monotonicity. *CoRR* abs/1708.07428 (2017). <https://arxiv.org/abs/1708.07428>
18. Wotzlaw, A., Speckenmeyer, E., Porschen, S.: Generalized k-ary tanglegrams on level graphs: a satisfiability-based approach and its evaluation. *Discrete Appl. Math.* **160**(16–17), 2349–2363 (2012). <https://doi.org/10.1016/j.dam.2012.05.028>

Non-crossing Paths with Geographic Constraints

Rodrigo I. Silveira¹(✉), Bettina Speckmann², and Kevin Verbeek²

¹ Department de Matemàtiques, Universitat Politècnica de Catalunya,
Barcelona, Spain

`rodrigo.silveira@upc.edu`

² Department of Mathematics and Computer Science, TU Eindhoven,
Eindhoven, The Netherlands

`{b.speckmann,k.a.b.verbeek}@tue.nl`

Abstract. A *geographic network* is a graph whose vertices are restricted to lie in a prescribed region in the plane. In this paper we begin to study the following fundamental problem for geographic networks: can a given geographic network be drawn without crossings? We focus on the seemingly simple setting where each region is a unit length vertical segment, and one wants to connect pairs of segments with a path that lies inside the convex hull of the two segments. We prove that when paths must be drawn as straight line segments, it is NP-complete to determine if a crossing-free solution exists. In contrast, we show that when paths must be monotone curves, the question can be answered in polynomial time. In the more general case of paths that can have any shape, we show that the problem is polynomial under certain assumptions.

1 Introduction

Highway, train, and river networks, airline and VLSI routing maps, information flow over the internet, and the flow of goods and people between different regions all have one thing in common: they can be effectively visualized as a *geographic network*: a graph, whose embedding is fixed, but not completely. The vertices of a geographic network are restricted to lie in a pre-scribed region while the edges might or might not be required to follow a particular course. In this paper we begin to study the following fundamental problem for geographic networks: can a given geographic network be drawn without crossings?

Many different formulations of this problem exist, which differ in aspects like the shape of the regions, the type of curve used to draw edges, and the type of graph being drawn. We study the seemingly simple variant where each region is a unit length vertical segment. We restrict the edges to be drawn to be simple curves that lie inside the convex hull of the vertical segments corresponding to the end vertices, to force edges to be more or less straight. Formally, we are given a graph $G = (V, E)$ and one unit vertical segment region I_v for each vertex $v \in V$. For each edge $(u, v) \in E$, we define the *tube* T_{uv} of (u, v) as the convex hull of $I_u \cup I_v$. The goal is to determine if it is possible to draw each vertex $v \in V$

as a point $p_v \in I_v$, and each edge $e = (u, v) \in E$ as a path from p_u to p_v that is contained in T_{uv} , such that no two paths cross at a point interior to both.

Related work. Force-directed layout methods for some particular cases of the problem studied here have been proposed in [1]. Also related is recent work on fitting planar graphs to planar maps [2], which is closely related to c-planarity for clustered graphs [5]. In the context of data imprecision, it has been shown that if the regions are vertical line segments or scaled copies of an arbitrary region, and the paths are straight line segments, determining if one can draw a cycle without crossings is NP-hard [10].

Another related problem studied is that of non-crossing matchings where each edge connects a point to a geometric object or a set of points. It was shown in [3] that the problem is polynomial in some special cases, most notably when matching a point to one of two other points, and NP-hard when the number of options increases. In particular, [3] shows that our problem for arbitrary (non-unit) vertical segment regions and straight-line segment paths is NP-hard, a fact that was also proven earlier in the Master's thesis of one of the authors of the current paper [12]. The same problem with unit-size square regions, but drawing general planar graphs instead of matchings, was also shown to be NP-hard in [4]. Considering monotone paths instead of straight-line paths, our problem is similar to the problem studied in [6], where the goal is to connect points with non-crossing paths that are rectilinear and xy -monotone. This problem was shown to be NP-hard recently [7]. However, our problem is slightly different, since endpoints are not fixed and paths are restricted to the tubes.

The most relevant previous work in our context is that on the *non-crossing connector problem* [9]: given m sets of points P_i , $1 \leq i \leq m$, and a region R_i (with $P_i \subset R_i$) for each i , the goal is to compute one curve inside each region R_i that goes through all the points in P_i and no two curves cross. It was shown in [9] that non-crossing connectors always exist if the regions are pseudo-disks. If that is not the case, existence can be decided in polynomial time for a few cases, while in general the problem is NP-complete. An important difference with our setting is that *all* given points P_i in each region must be connected.¹

Results and organization. We assume that G is a matching, so we can solely focus on drawing the edges. We then study the problem for different restrictions on the path representing the edges. In Sect. 2 we show that the problem is NP-complete if the paths must be straight-line segments. In Sect. 3 we show that, if paths must be x -monotone curves, we can decide in polynomial time if a crossing-free drawing exists. For arbitrary paths we can provide such a polynomial-time algorithm only under certain assumptions, as shown in Sect. 4.

¹ Confusingly, [9] quotes one of the current authors as stating that our problem is NP-complete for monotone paths, which is incorrect. Also, [9] incorrectly claims that it was proven in [12] that the problem is NP-complete for unit segments (the reduction in [12] uses segments of several lengths).

2 Straight Line Paths

In this section we show that, if the edges must be drawn as straight line segments, the problem is NP-complete. Let $V = \{v_1, \dots, v_{2n}\}$ and let I_i be the unit length vertical segment associated with v_i . For convenience we assume that there is an edge between v_{2i-1} and v_{2i} ($1 \leq i \leq n$), and let T_i be the corresponding tube.

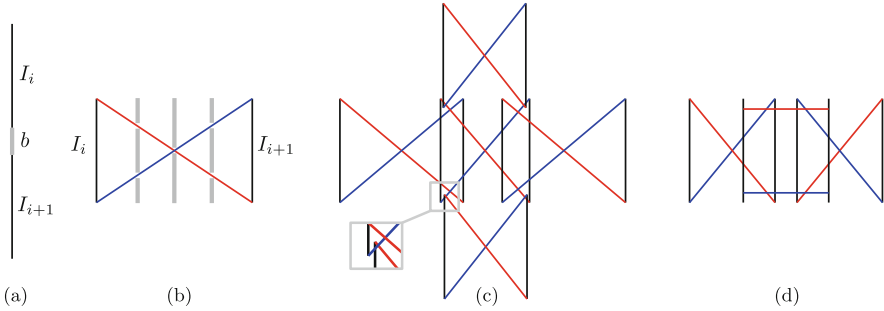
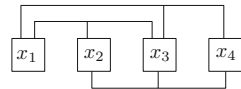


Fig. 1. Gadgets in the NP-hardness reduction. All black segments have unit length. (a) A blocker b . (b) A basic variable gadget, based on eight blockers (omitting for clarity the unit segments defining each of them). (c) Variable gadgets can be connected to propagate a truth value (blue or red diagonal). Note that blockers are not shown for clarity. (d) Negation of a variable. (Color figure online)

We prove NP-hardness by reduction from RECTILINEAR PLANAR 3-SAT [8]. An instance of this NP-complete problem consists of a 3-SAT formula and a rectilinear embedding of the graph associated to the formula. In the embedding all variable vertices lie on a straight line, and clauses are represented as horizontal lines with at most three vertical lines that connect to the variables appearing in the clause. See the figure on the right for an illustration of four variables and three clauses. The reduction relies on the following gadgets for variables and clauses.



Blockers. An essential building block is the construction of vertical edges that cannot be crossed by any segment in a solution, see Fig. 1(a). This is achieved by placing a tube connecting two disjoint vertical segments I_i and I_{i+1} exactly above each other,² forcing the segment between I_i and I_{i+1} to be part of any path connecting the tube.

Variable gadgets. The main component for modeling variables is the *basic gadget* shown in Fig. 1(b). Using a small set of blockers, we can limit the possible

² Note that the degenerate situation of two equal x -coordinates can be avoided by using small perturbations. The same applies to the other gadgets that make use of collinearities: they can all be removed while preserving the behavior of the gadgets.

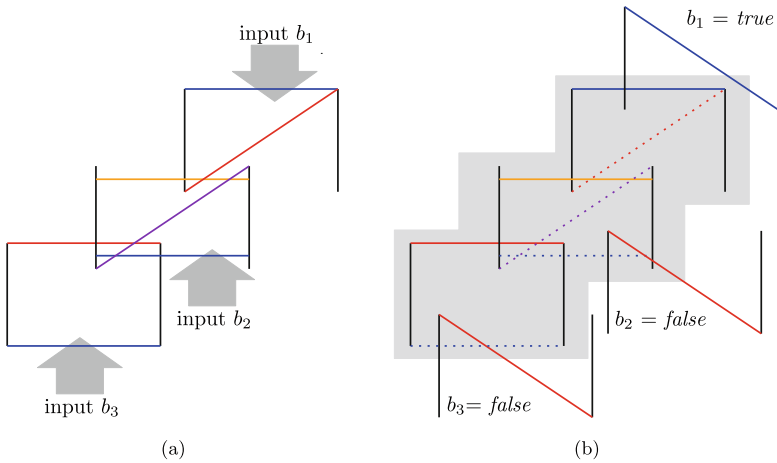


Fig. 2. Clause gadgets. (a) The gadget consists of three tubes. The three literals $\{b_1, b_2, b_3\}$ that participate in the clause cross the blue edges when their value is *false*. (b) Example of the only non-crossing solution when the first literal is *true* and the other two are *false* (dotted edges create crossings). (Color figure online)

connections for a tube to only two, shown in blue and red in the figure. These two solutions will correspond to the truth values *true* or *false* of the variable or literal. In general, if we want to limit the possible connections for a tube to a constant number of options, we can enforce this using a constant number of blockers. One generic way to achieve this is to choose three vertical segments (at arbitrary x -coordinates) spanning the tube and let these be interrupted by the chosen possible connections. In a non-degenerate situation this will leave only the chosen possible connections as options. As shown in Fig. 1(c), several basic gadgets can be connected in order to propagate the value in any of four directions. The value of a variable can be negated by adding a tube with two horizontal segments as options, as shown in Fig. 1(d).

Clause gadgets. In the embedding given in RECTILINEAR 3-SAT, a clause is represented by a horizontal line segment with three vertical segments, which connect to the variables. A horizontal segment can be recreated by using a single tube wide enough. Vertical segments can be represented by a chain of vertically stacked tubes (see Fig. 1(c)). The most interesting part of the clause is the point at which the three paths connect, in which the values of the three literals interact. In our gadget, this is achieved by using three tubes, as shown in Fig. 2. The top and bottom tubes have only two possible paths connecting them (for clarity, in the figures we omit the blockers needed to force this situation). The middle tube can be connected with three different edges. The three literals that form the clause attach to it through the blue edges. More precisely, a literal will have an edge crossing with one of the blue edges of the clause if and only if its value is *false*. The key property of the clause gadget is that there exist non-crossing paths

connecting the three tubes if and only if at least one literal is *true*. Note that the variable gadgets do not all connect to the clause gadgets from the bottom. However, we can easily achieve this construction by minor modifications to the rectilinear embedding and using the construction in Fig. 1(c).

With this construction we obtain the desired NP-hardness reduction: a satisfying truth assignment for the variables in the 3-SAT formula exists if and only if all tubes can be connected without crossings. In the full version [11] we show that the problem is in NP, leading to the following result.

Theorem 1. *Given n tubes defined by unit vertical segments, deciding if the tubes can be connected with straight line segments is NP-Complete.*

3 Monotone Paths

In this section we consider edges drawn as x -monotone paths. However, we first make some observations that hold for arbitrary paths.

We say that two tubes *fully cross* if the vertical segments are completely disjoint from the other tube, and the intersection of the two tubes is nonempty. The first basic observation is that whenever two tubes fully cross, no solution can exist. Therefore we assume from now on that no two tubes fully cross. The most interesting cases occur when two tubes intersect, without fully crossing. This necessarily happens because (at least) one of the vertical segments of a tube intersects the other tube. Figure 3 shows examples of such situations. We distinguish between *single intersections*, where only one tube segment intersects another tube, or *double intersections*, where two different segments intersect another tube (either both from the same tube, or one from each).

Single intersections (locally) induce a vertical order between the paths in any solution. For instance, in the situations in Fig. 3(a), the red tube can be considered *above* the blue one, because in any solution the red path will be above the blue one at the x -coordinate equal to the vertical segment creating the intersection. On the other hand, no such order exists for a double intersection. Indeed, in any double intersection there are solutions with both orders of the paths in the tubes, see Fig. 3(b). Based on this we define the *order graph*.

Order graph. The order graph of a set of tubes has a vertex for each tube and a directed edge from T_1 to T_2 if T_1 and T_2 have a single intersection where T_2 is above T_1 . We also add a directed edge from T_1 to T_2 if $T_1 \cap T_2 = \emptyset$ and T_1

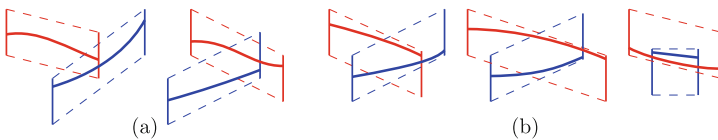


Fig. 3. Examples of tube intersections and solutions: (a) single and (b) double. Double intersections also admit solutions with the inverse red/blue order. (Color figure online)

and T_2 share an x -coordinate where T_2 is above T_1 . If T_1 and T_2 have a double intersection, we add an undirected edge between them. The order graph encodes enough information to decide whether a solution exists using x -monotone paths.

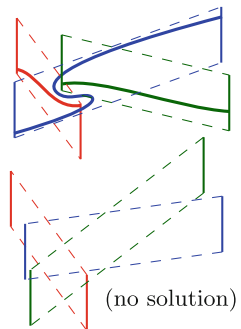
Theorem 2. *Given a set of tubes defined by unit vertical segments, the tubes can be connected with x -monotone paths if and only if the order graph contains no cycles of directed edges and no two tubes fully cross.*

Proof. First we prove that if the order graph has no cycle, and no two tubes fully cross, then there exists a solution. The directed edges in the order graph induce a partial order that can be extended to a total order on the tubes. Let T_1, \dots, T_n be that order from bottom to top. Let ℓ_i denote the bottom side of tube T_i . We maintain the following invariant: every drawn path p_i of tube T_i consists of parts of ℓ_j with $1 \leq j \leq i$ and vertical segments. We can clearly draw p_1 along ℓ_1 . Suppose we want to draw path p_i ($i > 1$). We start p_i at the highest intersection of the left vertical segment with any path p_j ($j < i$), or at ℓ_i if no such path exists. We follow a restricting path p_j until the right side of T_j , after which we drop down vertically, hitting either another path p_k or ℓ_i . In the latter case, or if we already hit ℓ_i before reaching the right side of T_j while following p_j , we can follow ℓ_i until hitting another restricting path. We then repeat this process until we reach the right side of T_i . The resulting path p_i only follows paths p_j ($j < i$), vertical segments, and ℓ_i , and thus satisfies the invariant. Finally note that p_i can leave T_i only if it is restricted by a path p_j intersecting the top of T_i . By the invariant, there must be some ℓ_k ($k < i$) intersecting the top of T_i , violating the order. We defer the proof of the remaining direction to the full version of the paper [11]. □

Therefore, the problem for monotone paths can be solved in polynomial time.

4 Arbitrary Paths

If we allow edges to be drawn by arbitrary paths, then a cycle in the order graph can sometimes be realized, as shown in the figure to the right (top)—the cycle here is red \rightarrow green \rightarrow blue \rightarrow red. However, that is not always the case, as the bottom example of the figure shows.



Nevertheless, if we disallow double intersections, then we can still decide in polynomial time whether a solution exists. The key idea is to use a result in [9] that shows that if the regions (in our case, tubes) form a set of pseudo-disks, then there is always a solution. Two tubes with a single intersection may not be pseudo-disks, but we can try to convert them into pseudo-disks by cutting off parts that cannot be used in any solution. This leads to a procedure that allows us to determine if a solution exists in polynomial time. The proof of the following is deferred to the full version [11].

Theorem 3. *Given a set of tubes defined by unit vertical segments such that no two tubes form a double intersection, one can determine if all the tubes can be connected without crossings using arbitrary paths in polynomial time.*

If we allow double intersections, the problem remains open. We finish by conjecturing that this problem admits a Helly-type property, implying a polynomial time algorithm.

Conjecture 1. If a set of tubes defined by unit vertical segments does not admit a solution with arbitrary paths, then there exists a constant size subset of tubes that also does not admit a solution.

Acknowledgements. R.I.S. was partially supported by projects MTM2015-63791-R (MINECO/FEDER) and Gen. Cat. DGR2014SGR46, and by MINECO's Ramón y Cajal program. B.S. and K.V. are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.023.208 and 639.021.541, respectively.


References

1. Abellanas, M., Aiello, A., Hernández, G., Silveira, R.I.: Network drawing with geographical constraints on vertices. In: Actas XI Encuentros de Geometría Computacional, pp. 111–118 (2005)
2. Alam, M.J., Kaufmann, M., Kobourov, S.G., Mchedlidze, T.: Fitting planar graphs on planar maps. *J. Graph Algorithms Appl.* **19**, 413–440 (2015). <https://doi.org/10.7155/jgaa.00367>
3. Aloupis, G., Cardinal, J., Collette, S., Demaine, E.D., Demaine, M.L., Dulieu, M., Fabila-Monroy, R., Hart, V., Hurtado, F., Langerman, S., Saumell, M., Seara, C., Taslakian, P.: Non-crossing matchings of points with geometric objects. *Comput. Geom.* **46**(1), 78–92 (2013). <https://doi.org/10.1016/j.comgeo.2012.04.005>
4. Angelini, P., Da Lozzo, G., Di Bartolomeo, M., Di Battista, G., Hong, S.-H., Patrignani, M., Roselli, V.: Anchored drawings of planar graphs. In: Duncan, C., Symvonis, A. (eds.) GD 2014. LNCS, vol. 8871, pp. 404–415. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45803-7_34
5. Feng, Q.-W., Cohen, R.F., Eades, P.: Planarity for clustered graphs. In: Spirakis, P. (ed.) ESA 1995. LNCS, vol. 979, pp. 213–226. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60313-1_145
6. Katz, B., Krug, M., Rutter, I., Wolff, A.: Manhattan-geodesic embedding of planar graphs. In: Eppstein, D., Gansner, E.R. (eds.) GD 2009. LNCS, vol. 5849, pp. 207–218. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11805-0_21
7. Klemz, B., Rote, G.: Ordered level planarity and geodesic planarity. In: Abstracts 33rd European Workshop on Computational Geometry, pp. 269–272 (2017)
8. Knuth, D.E., Raghunathan, A.: The problem of compatible representatives. *SIAM J. Discrete Math.* **5**(3), 422–427 (1992). <https://doi.org/10.1137/0405033>
9. Kratochvíl, J., Ueckerdt, T.: Non-crossing connectors in the plane. In: Chan, T.-H.H., Lau, L.C., Trevisan, L. (eds.) TAMC 2013. LNCS, vol. 7876, pp. 108–120. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38236-9_11

10. Löffler, M.: Existence and computation of tours through imprecise points. *Int. J. Comput. Geom. Appl.* **21**(01), 1–24 (2011). <https://doi.org/10.1142/S0218195911003524>
11. Silveira, R.I., Speckmann, B., Verbeek, K.: Non-crossing paths with geographic constraints (2017) [arXiv:1708.05486](https://arxiv.org/abs/1708.05486). <http://arxiv.org/abs/1708.05486>
12. Verbeek, K.: Non-crossing paths with fixed endpoints. Master's thesis, Technical University of Eindhoven (2008)

Special Representations

Planar L-Drawings of Directed Graphs

Steven Chaplick¹, Markus Chimani², Sabine Cornelsen³,
Giordano Da Lozzo⁴(✉), Martin Nöllenburg⁵, Maurizio Patrignani⁶,
Ioannis G. Tollis⁷, and Alexander Wolff¹ 

¹ Universität Würzburg, Würzburg, Germany
steven.chaplick@uni-wuerzburg.de

² Universität Osnabrück, Osnabrück, Germany
markus.chimani@uni-osnabrueck.de

³ Universität Konstanz, Konstanz, Germany
sabine.cornelsen@uni-konstanz.de

⁴ University of California, Irvine, CA, USA
gdalozzo@uci.edu

⁵ TU Wien, Vienna, Austria
noellenburg@ac.tuwien.ac.at

⁶ Roma Tre University, Rome, Italy
patrigna@dia.uniroma3.it

⁷ University of Crete, Heraklion, Greece
tollis@csd.uoc.gr

Abstract. We study planar drawings of directed graphs in the L-drawing standard. We provide necessary conditions for the existence of these drawings and show that testing for the existence of a planar L-drawing is an NP-complete problem. Motivated by this result, we focus on upward-planar L-drawings. We show that directed st-graphs admitting an upward- (resp. upward-rightward-) planar L-drawing are exactly those admitting a bitonic (resp. monotonically increasing) st-ordering. We give a linear-time algorithm that computes a bitonic (resp. monotonically increasing) st-ordering of a planar st-graph or reports that there exists none.

1 Introduction

In an *L-drawing* of a directed graph each vertex v is assigned a point in the plane with exclusive integer x - and y -coordinates, and each directed edge (u, v) consists of a vertical segment exiting u and of a horizontal segment entering v [1]. The drawings of two edges may cross and partially overlap, following the model

This research was initiated at the Bertinoro Workshop on Graph Drawing 2017. This article reports on work supported by the U.S. Defense Advanced Research Projects Agency (DARPA) under agreement no. AFRL FA8750-15-2-0092. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. This research was also partially supported by MIUR project “MODE – MORphing graph Drawings Efficiently”, prot. 20157EFM5C.001.

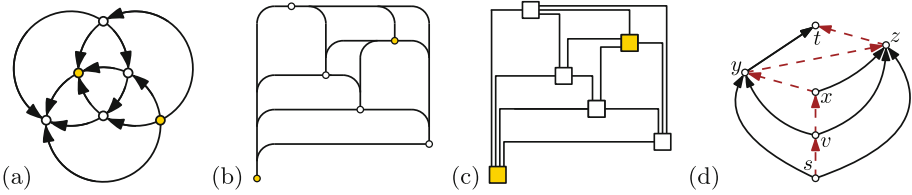


Fig. 1. (a) A bitonic st-orientation of the octahedron that admits an upward planar L-drawing (b). (c) The corresponding drawing in the Kandinsky model. (d) An upward planar st-graph U that does not admit an upward-planar L-drawing

of [17]. The ambiguity among crossings and bends is resolved by replacing bends with small rounded junctions. An L-drawing in which edges possibly overlap, but do not cross, is a *planar L-drawing*; see, e.g., Fig. 1b. A planar L-drawing is *upward planar* if its edges are y -monotone, and it is *upward-rightward planar* if its edges are simultaneously x -monotone and y -monotone.

Planar L-drawings correspond to drawings in the Kandinsky model [12] with exactly one bend per edge and with some restrictions on the angles around each vertex; see Fig. 1c. It is NP-complete [4] to decide whether a multigraph has a planar embedding that allows a Kandinsky drawing with at most one bend per edge [5]. On the other hand, every simple planar graph has a Kandinsky drawing with at most one bend per edge [5]. Bend-minimization in the Kandinsky-model is NP-complete [4] even if a planar embedding is given, but can be approximated by a factor of two [2, 11]. Heuristics for drawings in the Kandinsky model with empty faces and few bends have been discussed by Bekos et al. [3].

Bitonic st-orderings were introduced by Gronemann for undirected planar graphs [14] as an alternative to canonical orderings. They were recently extended to directed plane graphs [16]. In a bitonic st-ordering the successors of any vertex must form an increasing and then a decreasing sequence in the given embedding. More precisely, a *planar st-graph* is a directed acyclic graph with a single source s and a single sink t that admits a planar embedding in which s and t lie on the boundary of the same face. A planar st-graph always admits an upward-planar straight-line drawing [8]. An *st-ordering* of a planar st-graph is an enumeration π of the vertices with distinct integers, such that $\pi(u) < \pi(v)$ for every edge $(u, v) \in E$. Given a *plane st-graph*, i.e., a planar st-graph with a fixed upward-planar embedding \mathcal{E} , consider the list $S(v) = \langle v_1, v_2, \dots, v_k \rangle$ of successors of v in the left-to-right order in which they appear around v . The list $S(v)$ is *monotonically decreasing* with respect to an st-ordering π if $\pi(v_i) > \pi(v_{i+1})$ for $i = 1, \dots, k - 1$. It is *bitonic* with respect to π if there is a vertex v_h in $S(v)$ such that $\pi(v_i) < \pi(v_{i+1})$, $i = 1, \dots, h - 1$ and $\pi(v_i) > \pi(v_{i+1})$, $i = h, \dots, k - 1$. For an upward-planar embedding \mathcal{E} , an st-ordering π is *bitonic* or *monotonically decreasing*, respectively if the successor list of each vertex is bitonic or monotonically decreasing, respectively. Here, $\langle \mathcal{E}, \pi \rangle$ is called a *bitonic pair* or *monotonically decreasing pair*, respectively, of G .

Gronemann used bitonic st-orderings to obtain on the one hand upward-planar polyline grid drawings in quadratic area with at most $|V| - 3$ bends in total [16] and on the other hand contact representations with upside-down oriented T-shapes [15]. A bitonic st-ordering for biconnected undirected planar graphs can be computed in linear time [14] and the existence of a bitonic st-ordering for plane (directed) st-graphs can also be decided in linear time [16]. However, in the variable embedding scenario no algorithm is known to decide whether an st-graph G admits a bitonic pair. Bitonic st-orderings turn out to be strongly related to upward-planar L-drawings of st-graphs. In fact, the y -coordinates of an upward-planar L-drawing yield a bitonic st-ordering.

In this work, we initiate the investigation of planar and upward-planar L-drawings. In particular, our contributions are as follows. (i) We prove that deciding whether a directed planar graph admits a planar L-drawing is NP-complete. (ii) We characterize the planar st-graphs admitting an upward (upward-rightward, resp.) planar L-drawing as the st-graphs admitting a bitonic (monotonic decreasing, resp.) st-ordering. (iii) We provide a linear-time algorithm to compute an embedding, if any, of a planar st-graph that allows for a bitonic st-ordering. This result complements the analogous algorithm proposed by Gronemann for undirected graphs [14] and extends the algorithm proposed by Gronemann for planar st-graphs in the fixed embedding setting [16]. (iv) Finally, we show how to decide efficiently whether there is a planar L-drawing for a plane directed graph with a fixed assignment of the edges to the four ports of the vertices.

Due to space limitations, full proofs are provided in [6].

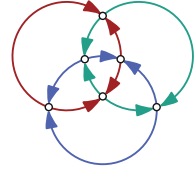
2 Preliminaries

We assume familiarity with basic graph drawing concepts and in particular with the notions of connectivity and SPQR-trees (see also [6,9]).

A (simple, finite) directed graph $G = (V, E)$ consists of a finite set V of vertices and a finite set $E \subseteq \{(u, v) \in V \times V; u \neq v\}$ of ordered pairs of vertices. If (u, v) is an edge then v is a *successor* of u and u is a *predecessor* of v . A graph is *planar* if it admits a drawing in the plane without edge crossings. A *plane graph* is a planar graph with a fixed *planar embedding*, i.e., with fixed circular orderings of the edges incident to each vertex—determined by a planar drawing—and with a fixed outer face.

Given a planar embedding and a vertex v , a pair of consecutive edges incident to v is *alternating* if they are not both incoming or both outgoing. We say that v is *k-modal* if there exist exactly k alternating pairs of edges in the cyclic order around v . An embedding of a directed graph G is *k-modal*, if each vertex is at most k -modal. A 2-modal embedding is also called *bimodal*. An upward-planar drawing determines a bimodal embedding. However, the existence of a bimodal embedding is not a sufficient condition for the existence of an upward-planar drawing. Deciding whether a directed graph admits an upward-planar (straight-line) drawing is an NP-hard problem [13].

L-Drawings. A planar L-drawing determines a 4-modal embedding. This implies that there exist planar directed graphs that do not admit planar L-drawings. A 6-wheel whose central vertex is incident to alternating incoming and outgoing edges is an example of a graph that does not admit any 4-modal embedding, and therefore any planar L-drawing.



On the other hand, the existence of a 4-modal embedding is not sufficient for the existence of a planar L-drawing. E.g., the octahedron depicted in the figure on the right does not admit a planar L-drawing. Since the octahedron is triconnected, it admits a unique combinatorial embedding (up to a flip). Each vertex is 4-modal. However, the rightmost vertex in a planar L-drawing must be 1-modal or 2-modal.

Any upward-planar L-drawing of an st-graph G can be modified to obtain an upward-planar drawing of G : Redraw each edge as a y -monotone curve arbitrarily close to the drawing of the corresponding 1-bend orthogonal polyline while avoiding crossings and edge-edge overlaps. However, not every upward-planar graph admits an upward-planar L-drawing. E.g., the graph in Fig. 1d contains a subgraph that does not admit a bitonic st-ordering [16]. In Sect. 4 (Theorem 3), we show that this means it does not admit an upward planar L-drawing.

The Kandinsky Model. In the Kandinsky model [12], vertices are drawn as squares of equal sizes on a grid and edges—usually undirected—are drawn as orthogonal polylines on a finer grid; see Fig. 1c. Two consecutive edges in the clockwise order around a vertex define a face and an angle in $\{0, \pi/2, \pi, 3\pi/2, 2\pi\}$ in that face. In order to avoid edges running through other vertices, the Kandinsky model requires the so called *bend-or-end property*: There is an assignment of bends to vertices with the following three properties. (a) Each bend is assigned to at most one vertex. (b) A bend may only be assigned to a vertex to which it is connected by a segment (i.e., it must be the first bend on an edge). (c) If e_1, e_2 are two consecutive edges in the clockwise order around a vertex v that form a 0 angle inside face f , then a bend of e_1 or e_2 forming a $3\pi/2$ angle inside f must be assigned to v . Further, the Kandinsky model requires that there are no *empty faces*.

Given a planar L-drawing, consider a vertex v and all edges incident to one of the four ports of v . By assigning to v all bends on these edges—except the bend furthest from v —we satisfy the bend-or-end property. This implies the following lemma, which is proven in [6].

Lemma 1. *A graph has a planar L-drawing if and only if it admits a drawing in the Kandinsky model with the following properties: (i) Each edge bends exactly once; (ii) at each vertex, the angle between any two outgoing (or between any two incoming) edges is 0 or π ; and (iii) at each vertex, the angle between any incoming edge and any outgoing edge is $\pi/2$ or $3\pi/2$.*

3 General Planar L-Drawings

We consider the problem of deciding whether a graph admits a planar L-drawing. In Sect. 3.1, we show that the problem is NP-complete if no planar embedding

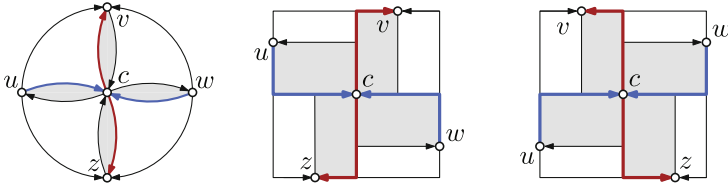


Fig. 2. 4-wheel graph W and two planar L-drawings of W .

is given. In the fixed embedding setting (Sect. 3.2) the problem can be described as an ILP. It is solvable in linear time if we also fix the ports.

3.1 Variable Embedding Setting

As a central building block for our hardness reduction we use a directed graph W that can be constructed starting from a 4-wheel with central vertex c and rim (u, v, w, z) . We orient the edges of W so that v and z (the V -ports of W) are sinks and u and w (the H -ports of W) are sources. Finally, we add directed edges (v, c) , (z, c) , (c, w) , and (c, u) ; see Fig. 2. We now provide Lemma 2 which describes the key property of planar L-drawings of W .

Lemma 2. *In any planar L-drawing of W with cycle (u, v, w, z) as the outer face the edges of the outer face form a rectangle (that contains vertex c).*

We are now ready to give the main result of the section.

Theorem 1. *It is NP-complete to decide whether a directed graph admits a planar L-drawing.*

Sketch of proof. We reduce from the NP-complete problem of HV-rectilinear planarity testing [10]. In this problem, the input is a biconnected degree-4 planar graph G with edges labeled either H or V, and the goal is to decide whether G admits an HV-drawing, i.e., a planar drawing such that each H-edge (V-edge) is drawn as a horizontal (vertical) segment. Starting from G , we construct a graph G' by replacing: (i) vertices with 4-wheels as in Fig. 2; (ii) V-edges with the gadget shown in Fig. 3a; and (iii) H-edges with an appropriately rotated and re-oriented version of the V-edge gadget. If (u, v) is a V-edge, the two vertices labeled u and v of its gadget are identified with a V-port of the respective vertex gadgets. Otherwise, they are identified with an H-port. Figure 3b shows a vertex gadget with four incident edges. The proof that G' and G are equivalent is somewhat similar to Brückner’s hardness proof in [5, Theorem 3] and exploits Lemma 2. Refer to [6] for the full details. □

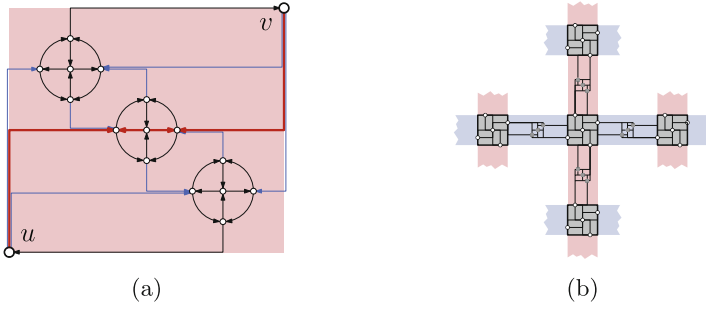


Fig. 3. (a) Edge gadget for a V-edge. (b) Connections among gadgets.

3.2 Fixed Embedding and Port Assignment

In this section, we show how to decide efficiently whether there is a planar L-drawing for a plane directed graph with a fixed assignment of the edges to the four ports of the vertices. Using Lemma 1 and the ILP formulation of Barth et al. [2], we first set up linear inequalities that describe whether a plane 4-modal graph has a planar L-drawing. Using these inequalities, we then transform our decision problem into a matching problem that can be solved in linear time.

We call a vertex v an *in/out-vertex* on a face f if v is incident to both, an incoming edge and an outgoing edge on f . Let $x_{vf} \in \{0, 1, 2\}$ describe the angle in a face f at a vertex v : the angle between two outgoing or two incoming edges is $x_{vf} \cdot \pi$ and the angle between an incoming and an outgoing edge is $x_{vf} \cdot \pi + \pi/2$. Let $x_{fe}^v \in \{0, 1\}$ be 1 if there is a convex bend in face f on edge e assigned to a vertex v to fulfill the bend-or-end property. There is a planar L-drawing with these parameters if and only if the following four conditions are satisfied (see [6] for details): (1) The angles around a vertex v sum to 2π . (2) Each edge has exactly one bend. (3) The number of convex angles minus the number of concave angles is 4 in each inner face and -4 in the outer face. (4) The bend-or-end property is fulfilled, i.e., for any two edges e_1 and e_2 that are consecutive around a vertex v and that are both incoming or both outgoing, and for the faces f_1 , f , and f_2 that are separated by e_1 and e_2 (in the cyclic order around v), it holds that $x_{vf} + x_{f_1e_1}^v + x_{f_2e_2}^v \geq 1$. Let $e = (v, w)$ be incident to faces f and h , Condition (2) implies $-x_{he}^v - x_{he}^w = x_{fe}^v + x_{fe}^w - 1$. Hence, (3) yields

$$(3') \sum_{e=(v,w) \text{ incident to } f} (x_{fe}^v + x_{fe}^w) - \sum_{v \text{ on } f} x_{vf} = \pm 2 + (\# \text{ in/out-vertices on } f - \deg f)/2.$$

Observe that the number of in/out-vertices on a face f is odd if and only if $\deg f$ is odd. Moreover, if we omit the bend-or-end property, we can formulate the remaining conditions as an uncapacitated network flow problem. The network has three types of nodes: one for each vertex, face, and edge of the graph. It has two types of edges: from vertices to incident faces and

from faces to incident edges. The supplies are $\lceil \frac{4-k}{2} \rceil$ for the k -modal vertices, $\pm 2 + 1/2 \cdot (\#in/out\text{-vertices} - \deg f)$ for a face f , and -1 for the edges.

Theorem 2. *Given a directed plane graph G and labels $out(e) \in \{\text{top, bottom}\}$ and $in(e) \in \{\text{right, left}\}$ for each edge e , it can be decided in linear time whether G admits a planar L-drawing in which each edge e leaves its tail at $out(e)$ and enters its head at $in(e)$.*

Sketch of proof. First, we have to check whether the cyclic order of the edges around a vertex is compatible with the labels. The labels determine the bends and the angles around the vertices, i.e., $x_{fe}^v + x_{fe}^w$ for each edge $e = (v, w)$ and each incident face f , and x_{vf} for each vertex v and each incidence to a face f . We check whether these values fulfill Conditions 1, 2, and 3'. In order to also check Condition 4, we first assign for each port of a vertex v , all but the middle edges to v (where a *middle edge* of a port is the last edge in clockwise order bending to the left or the first edge bending to the right). We check whether we thereby assign an edge more than once. Assigning the middle edges can be reduced to a matching problem in a bipartite graph of maximum degree 2 where the nodes on one side are the ports with two middle edges and the nodes on the other side are the unassigned edges. □

4 Upward- and Upward-Rightward Planar L-Drawings

In this section, we characterize (see Theorem 3) and construct (see Theorem 6) upward-planar and upward-rightward planar L-drawings.

4.1 A Characterization via Bitonic st-Orderings

Characterizing the plane directed graphs that admit an L-drawing is an elusive goal. However, we can characterize two natural subclasses of planar L-drawings via bitonic st-orderings.

Theorem 3. *A planar st-graph admits an upward- (upward-rightward-) planar L-drawing if and only if it admits a bitonic (monotonically decreasing) pair.*

Sketch of proof. “ \Rightarrow ”: Let $G = (V, E)$ be an st-graph with n vertices. The y -coordinates of an upward- (upward-rightward-) planar L-drawing of G yield a bitonic (monotonically decreasing) st-ordering.

“ \Leftarrow ”: Given a bitonic (monotonically decreasing) st-ordering π of $G = (V, E)$, we construct an upward- (upward-rightward-) planar L-drawing of G using an idea of Gronemann [16]. For each vertex v , we use $\pi(v)$ as its y -coordinate.

For the x -coordinates we use a linear extension of a partial order \prec . Let v_1, \dots, v_n be the vertices of G in the ordering given by π . Let G_i be the subgraph of G induced by $V_i = \{v_1, \dots, v_i\}$. To construct \prec , we augment G_i to \overline{G}_i in such a way that the outer face $f_{\overline{G}_i}$ of \overline{G}_i is a simple cycle and all vertices on $f_{\overline{G}_i}$ are comparable: We start with a triangle on v_1 and two new vertices v_{-1} and

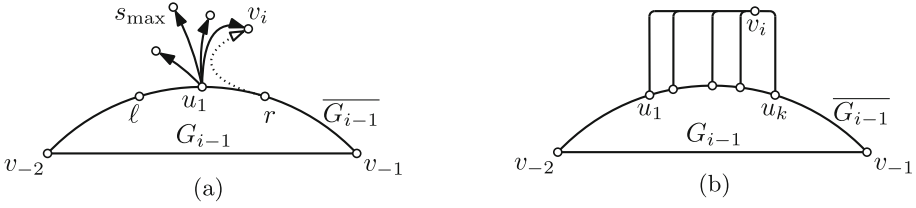


Fig. 4. How to turn a bitonic st-ordering into a planar L-drawing.

v_{-2} , with y -coordinates -1 and -2 , respectively, and set $v_{-2} \prec v_1 \prec v_{-1}$. For $i = 2, \dots, n$, let u_1, \dots, u_k be the predecessors of v_i in ascending order with respect to \prec . If π is monotonically decreasing or if $k = 1$, we add an edge e with head v_i . The tail of e is the right neighbor r of u_k or the left neighbor ℓ of u_1 on $f_{\overline{G}_i}$, respectively, if the maximum successor s_{\max} of u_1 is to the left (or equal to) or the right of v_i , respectively; see Fig. 4a. Now let u_1, \dots, u_k be the predecessors of v_i in the possibly augmented graph; see Fig. 4b. We add the condition $u_{k-1} \prec v_i \prec u_k$. \square

Corollary 1. Any undirected planar graph can be oriented such that it admits an upward-planar L-drawing.

Proof. Triangulate the graph G and construct a bitonic st-ordering for undirected graphs [14]. Orient the edges from smaller to larger st-numbers. \square

4.2 Bitonic st-Orderings in the Variable Embedding Setting

By Theorem 3, testing for the existence of an upward- (upward-rightward-) planar L-drawing of a planar st-graph G reduces to testing for the existence of a bitonic (monotonically decreasing) pair $\langle \mathcal{E}, \pi \rangle$ for G . In this section, we give a linear-time algorithm to test an st-graph for the existence of a bitonic pair $\langle \mathcal{E}, \pi \rangle$.

The following lemma is proved in [6].

Lemma 3. Let $G = (V, E)$ be a planar st-graph with source s , sink t , and $(s, t) \notin E$. Then there exists a supergraph $G' = (V', E')$ of G , where $V' = V \cup \{s'\}$ and $E' = E \cup \{(s', s), (s', t)\}$, such that (i) G' is an st-graph with source s' and sink t , and (ii) G' admits a bitonic (resp., monotonically increasing) st-ordering if and only if G does.

By Lemma 3, in the following we assume that an st-graph G always contains edge (s, t) . Hence, either G coincides with edge (s, t) , which trivially admits a bitonic st-ordering, or it is biconnected.

A path p from u to v in a directed graph is *monotonic increasing* (*monotonic decreasing*) if it is exclusively composed of forward (backward) edges. A path p is *monotonic* if it is either monotonic increasing or monotonic decreasing. A path p with endpoints u and v is *bitonic* if it consists of a monotonic increasing

path from u to w and of a monotonic decreasing path from w to v ; if $u \neq w$ and $v \neq w$, then the path p is *strictly bitonic* and w is the *apex* of p . An st-graph G is *v-monotonic*, *v-bitonic*, or *strictly v-bitonic* if the subgraph of G induced by the successors of v is, after the removal of possible transitive edges, a monotonic, bitonic, or strictly-bitonic path p , respectively. The apex of p , if any, is also called the *apex* of v in G . If p is monotonic and it is directed from u to w , then vertices u and w are the *first successor* of v in G and the *last successor* of v in G , respectively. If p is strictly bitonic, then its endpoints are the *first successors* of v in G . If p consists of a single vertex, then such a vertex is both the *first and the last successor* of v in G . Let G be an st-graph and let G^* be an st-graph obtained by augmenting G with directed edges. We say that the pair $\langle G, G^* \rangle$ is *v-monotonic*, *v-bitonic*, or *strictly v-bitonic* if the subgraph of G^* induced by the successors of v in G is, after the removal of possible transitive edges, a monotonic, bitonic, or strictly-bitonic path, respectively.

Although Gronemann [16] didn't state this explicitly, the following theorem immediately follows from the proof of his Lemma 4.

Theorem 4 ([16]). *A plane st-graph $G = (V, E)$ admits a bitonic st-ordering if and only if it can be augmented with directed edges to a planar st-graph G^* such that, for each vertex $v \in V$, the pair $\langle G, G^* \rangle$ is v-bitonic. Further, any st-ordering of G^* is a bitonic st-ordering of G .*

In the remainder of the section, we show how to test in linear-time whether it is possible to augment a biconnected st-graph G to an st-graph G^* in such a way that the pair $\langle G, G^* \rangle$ is v-bitonic, for any vertex v of G . By virtue of Theorem 4, this allows us to test the existence of a bitonic pair $\langle \mathcal{E}, \pi \rangle$ for G . We perform a bottom-up visit of the SPQR-tree T of G rooted at the reference edge (s, t) and show how to compute an augmentation for the pertinent graph of each node $\mu \in T$ together with an embedding of it, if any exists.

Note that each vertex in an st-graph is on a directed path from s to t . Further, by the choice of the reference edge, neither s nor t are internal vertices of the pertinent graph of any node of T . This leads to the next observation.

Observation 1. *For each node $\mu \in T$ with poles u and v , the pertinent graph $\text{pert}(\mu)$ of μ is an st-graph whose source and sink are u and v , or vice versa.*

Let e be a virtual edge of $\text{skel}(\mu)$ corresponding to a node ν whose pertinent graph is an st-graph with source s_ν and sink t_ν . By Observation 1, we say that e *exits* s_ν and *enters* t_ν .

The outline of the algorithm is as follows. Consider a node $\mu \in T$ and suppose that, for each child μ_i of μ , we have already computed a pair $\langle \text{pert}^*(\mu_i), \mathcal{E}_i^* \rangle$ such that $\text{pert}^*(\mu_i)$ is an augmentation of $\text{pert}(\mu_i)$, \mathcal{E}_i^* is an embedding of $\text{pert}^*(\mu_i)$, and $\langle \text{pert}(\mu_i), \text{pert}^*(\mu_i) \rangle$ is v-bitonic, for each vertex v of $\text{pert}(\mu_i)$. We show how to compute a pair $\langle \text{pert}^*(\mu), \mathcal{E}^* \rangle$ for node μ , such that (i) the pair $\langle \text{pert}(\mu), \text{pert}^*(\mu) \rangle$ is v-bitonic for each vertex v in $\text{pert}(\mu)$, and (ii) the restriction of \mathcal{E}^* to $\text{pert}^*(\mu_i)$ is \mathcal{E}_i^* , up to a flip. In the following, for the sake of

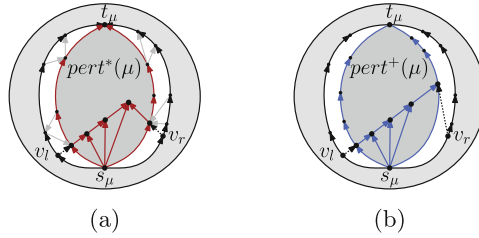


Fig. 5. Illustration for Lemma 4.

clarity, we first describe an overall quadratic-time algorithm. We will refine this algorithm to run in linear time at the end of the section.

For a node $\mu \in T$, we say that the pair $\langle \text{pert}(\mu), \text{pert}^*(\mu) \rangle$ is of Type B if it is strictly s_μ -bitonic and it is of Type M if it is s_μ -monotonic. For simplicity, we also say that node μ is of Type B or of Type M when, during the traversal of T , we have constructed an augmentation $\text{pert}^*(\mu)$ for μ such that $\langle \text{pert}(\mu), \text{pert}^*(\mu) \rangle$ is of Type B or of Type M, respectively. Figure 5 shows an example where an augmentation G^* of G contains an augmentation $\text{pert}^*(\mu)$ for μ which is replaced with an augmentation $\text{pert}^+(\mu)$ such that $\langle \text{pert}(\mu), \text{pert}^*(\mu) \rangle$ is of Type B, $\langle \text{pert}(\mu), \text{pert}^+(\mu) \rangle$ is of Type M, and G^* admits a bitonic st-ordering if and only if it still does after this replacement. The following lemma formally shows that this type of replacement is always possible.

Lemma 4. *Let G be a biconnected st-graph and let G^* be an augmentation of G such that $\langle G, G^* \rangle$ is v -bitonic, for each vertex v of G . Consider a node μ of the SPQR-tree of G and let $\text{pert}^*(\mu)$ be the subgraph of G^* induced by the vertices of $\text{pert}(\mu)$. Suppose that $\langle \text{pert}(\mu), \text{pert}^*(\mu) \rangle$ is of Type B and that $\text{pert}(\mu)$ also admits an augmentation $\text{pert}^+(\mu)$ such that $\langle \text{pert}(\mu), \text{pert}^+(\mu) \rangle$ is of Type M and it is v -bitonic, for each vertex v of $\text{pert}(\mu)$. There exists an augmentation G^+ of G such that $\langle G, G^+ \rangle$ is v -bitonic, for each vertex v of G , and such that the subgraph of G^+ induced by the vertices of $\text{pert}(\mu)$ is $\text{pert}^+(\mu)$.*

Consider a node μ of the SPQR-tree T of G . We now show how to test the existence of a pair $\langle \text{pert}^*(\mu), \mathcal{E}^* \rangle$ such that (i) μ is of Type M or, secondarily, of Type B, or report that no such a pair exists, and (ii) \mathcal{E}^* is a planar embedding of $\text{pert}^*(\mu)$. In fact, by Lemma 4, an embedding of μ of Type M would always be preferable to an embedding of Type B.

In any planar embedding \mathcal{E} of $\text{pert}(\mu)$ in which the poles are on the outer face f_{out} of \mathcal{E} , we call *left path* (*right path*) of \mathcal{E} the path that consists of the edges encountered in a clockwise traversal (in a counter-clockwise traversal) of the outer face of \mathcal{E} from s_μ to t_μ .

The following observation will prove useful to construct embedding \mathcal{E}^* .

Observation 2. *Let $\langle \text{pert}^*(\mu), \mathcal{E}^* \rangle$ be a pair such that $\langle \text{pert}(\mu), \text{pert}^*(\mu) \rangle$ is s_μ -bitonic and \mathcal{E}^* is a planar embedding of $\text{pert}^*(\mu)$ in which s_μ and t_μ lie on the external face. We have that:*

- (i) If μ is of Type M, then the first and the last successors of s_μ in $\text{pert}^*(\mu)$ lie one on the left path and the other on the right path of \mathcal{E}^* . In particular, if the first and the last successor of μ are the same vertex, then such a vertex belongs to both the left path and the right path of \mathcal{E}^* .
- (ii) If μ is of Type B, then the two first successors of s_μ in $\text{pert}^*(\mu)$ lie one on the left path and the other on the right path of \mathcal{E}^* .

We distinguish four cases based on whether node μ is an S-, P-, Q-, or R-node.

Q-node. Here, $\langle \text{pert}(\mu), \text{pert}(\mu) \rangle$ is trivially of Type M, i.e., $\text{pert}^*(\mu) = \text{pert}(\mu)$.

S-node. Let e_1, \dots, e_k be the virtual edges of $\text{skel}(\mu)$ in the order in which they appear from the source s_μ to the target t_μ of $\text{skel}(\mu)$, and let μ_1, \dots, μ_k be the corresponding children of μ , respectively. We obtain $\text{pert}^*(\mu)$ by replacing each virtual edge e_i in $\text{skel}(\mu)$ with $\text{pert}^*(\mu_i)$. Also, we obtain the embedding \mathcal{E}^* by arbitrarily selecting a flip for each embedding \mathcal{E}_i^* of $\text{pert}^*(\mu_i)$. Clearly, node μ is of Type M if and only if μ_1 is of Type M and it is of Type B, otherwise.

P-node. Let e_1, \dots, e_k be the virtual edges of $\text{skel}(\mu)$ and let μ_1, \dots, μ_k be the corresponding children of μ , respectively.

First, observe that if there exists more than one child of μ that is of Type B, then node μ does not admit an augmentation $\text{pert}^*(\mu)$ where $\langle \text{pert}(\mu), \text{pert}^*(\mu) \rangle$ is s_μ -bitonic. In fact, if there exist two such nodes μ_i and μ_j , then both the subgraphs of $\text{pert}^*(\mu_i)$ and $\text{pert}^*(\mu_j)$ induced by the successors of s_μ in $\text{pert}(\mu_i)$ and in $\text{pert}(\mu_j)$, respectively, contain an apex vertex. This implies that s_μ would have more than one apex.

Second, observe that if there exists a child μ_i of μ of Type B and the edge (s_μ, t_μ) belongs to $\text{pert}(\mu)$, then node μ does not admit an augmentation $\text{pert}^*(\mu)$ such that $\langle \text{pert}(\mu), \text{pert}^*(\mu) \rangle$ is s_μ -bitonic. In fact, $\text{pert}^*(\mu_i)$ contains an apex of s_μ different from t_μ ; this is due to the fact that edge $(s_\mu, t_\mu) \notin \text{pert}^*(\mu_i)$. Also, vertex t_μ must be an apex of s_μ in any augmentation $\text{pert}^*(\mu)$ of $\text{pert}(\mu)$ such that $\langle \text{pert}(\mu), \text{pert}^*(\mu) \rangle$ is v -bitonic, for each vertex v of $\text{pert}(\mu)$. Namely, any augmentation $\text{pert}^*(\mu)$ of $\text{pert}(\mu)$ yields an st-graph with source s_μ and sink t_μ and, as such, no directed path exits from t_μ in $\text{pert}^*(\mu)$. As for the observation in the previous paragraph, this implies that s_μ would have more than one apex.

We construct $\text{pert}^*(\mu)$ as follows. We embed $\text{skel}(\mu)$ in such a way that the edge (s_μ, t_μ) , if any, or the virtual edge corresponding to the unique child of μ that is of Type B, if any, is the right-most virtual edge in the embedding. Let e_1, \dots, e_k be the virtual edges of $\text{skel}(\mu)$ in the order in which they appear clockwise around s_μ in $\text{skel}(\mu)$. Then, for each child μ_i of μ , we choose a flip of embedding \mathcal{E}_i^* such that a first successor of s_μ in $\text{pert}^*(\mu_i)$ lies along the left path of \mathcal{E}_i^* . Now, for $i = 1, \dots, k - 2$, we add an edge connecting the last successor of s_μ in $\text{pert}^*(\mu_i)$ and the first successor of s_μ in $\text{pert}^*(\mu_{i+1})$. Finally, we possibly add an edge connecting the last successor v_l of s_μ in $\text{pert}^*(\mu_{k-1})$ and a suitable vertex in $\text{pert}^*(\mu_k)$. Namely, if a node μ_k is of Type B, then we add an edge between v_l and the first successor of s_μ in $\text{pert}^*(\mu_k)$ that lies along the left path of \mathcal{E}_k^* . If μ_k is of Type M and it is not a Q-node, then we add an edge between

v_l and the first successor of s_μ in $\text{pert}^*(\mu_k)$. Otherwise $\text{pert}^*(\mu_k) = (s_\mu, t_\mu)$ and we add the edge (v_l, t_μ) if no such an edge belongs to $\text{pert}^*(\mu_{k-1})$.

Observe that, the added edges do not introduce any directed cycle as there exists no directed path from a vertex in $\text{pert}^*(\mu_{i+1})$ to a vertex in $\text{pert}^*(\mu_i)$. Further, by Observation 2 the added edges do not disrupt planarity. Therefore, the obtained augmentation $\text{pert}^*(\mu)$ of $\text{pert}(\mu)$ is, in fact, a planar st-graph.

Finally, we have that node μ is of Type M if and only if μ_k is of Type M.

R-node. The case of an R-node μ is detailed in [6]. For each node v of $\text{skel}(\mu)$, we have to consider the virtual edges e_1, \dots, e_k of $\text{skel}(\mu)$ exiting v and the corresponding children μ_1, \dots, μ_k of μ , respectively. Similarly to the P-node case, we pursue an augmentation of $\text{pert}(\mu)$ by inserting edges that connect $\text{pert}(\mu_i)$ with $\text{pert}(\mu_{i+1})$, with $i = 1, \dots, k-1$. Differently from the P-node case, however, more than one $\text{pert}(\mu_i)$ may contain an edge between the poles of μ_i . Further, also the faces of $\text{skel}(\mu)$ may play a role, introducing additional constraints on the existence and the choice of the augmentation.

We have the following theorem.

Theorem 5. *It is possible to decide in linear time whether a planar st-graph G admits a bitonic pair $\langle \mathcal{E}, \pi \rangle$.*

Proof. Let ρ be the root of the SPQR-tree of G . The algorithm described above computes a pair $\langle \text{pert}^*(\rho), \mathcal{E}^* \rangle$ for G , if any exists, such that (i) the st-graph $\text{pert}^*(\rho)$ is an augmentation of G , (ii) for any vertex v of G , $\langle \text{pert}(\rho), \text{pert}^*(\rho) \rangle$ is v -bitonic, and (iii) \mathcal{E}^* is a planar embedding of $\text{pert}^*(\rho)$. Let \mathcal{E} be the restriction of \mathcal{E}^* to G . By Theorem 4, any st-ordering π of $\text{pert}^*(\rho)$ is a bitonic st-ordering of G with respect to \mathcal{E} . Hence, $\langle \mathcal{E}, \pi \rangle$ is a bitonic pair of G .

We first show that the described algorithm has a quadratic running time. Then, we show how to refine it in order to run in linear time. For each node μ of T , the algorithm stores a pair $\langle \text{pert}^*(\mu), \mathcal{E} \rangle$. Processing a node takes $O(|\text{pert}^*(\mu)|)$ time. Since $|\text{pert}^*(\mu)| \in O(|\text{pert}(\mu)|)$, the overall running time is $O(|G|^2)$.

To achieve a linear running time, observe that we do not need to compute the embeddings of the augmented pertinent graphs $\text{pert}^*(\mu)$, for each node μ of T , during the bottom-up traversal of T . In fact, any embedding \mathcal{E}^* of $\text{pert}^*(\rho)$ yields an embedding \mathcal{E} of G such that π is bitonic with respect to \mathcal{E} . To determine the endpoints of the augmenting edges, we only need to associate a constant amount of information with the nodes of T . Namely, for each node μ in T , we maintain (i) whether μ is of Type B or of Type M, (ii) if μ is of Type M, the first successor and the last successor of s_μ in $\text{pert}^*(\mu)$, and (iii) if μ is of Type B, the two first successors of s_μ in $\text{pert}^*(\mu)$. Therefore, processing a node takes $O(|\text{skel}(\mu)|)$ time. Since the sum of the sizes of the skeletons of the nodes in T is linear in the size of G [7], the overall running time is linear. □

Corollary 2. *It is possible to decide in linear time whether a planar st-graph G admits a monotonically decreasing pair $\langle \mathcal{E}, \pi \rangle$.*

Proof. The statement immediately follows from the fact that, in the algorithm described in this section, when computing a pair $\langle \text{pert}^*(\mu), \mathcal{E}^* \rangle$ for each node μ

in T , a pair $\langle \text{pert}(\mu), \text{pert}^*(\mu) \rangle$ of Type M is built whenever possible. Therefore, rejecting instances for which a pair $\langle \text{pert}(\mu), \text{pert}^*(\mu) \rangle$ of Type B is needed yields the desired algorithm. \square

In conclusion, we have the following main result.

Theorem 6. *It can be tested in linear time whether a planar st-graph admits an upward- (upward-rightward-) planar L-drawing, and if so, such a drawing can be constructed in linear time.*

Proof. We first test in linear time whether a planar st-graph admits a bitonic pair (Theorem 5) or a monotonically decreasing pair (Corollary 2). Then, Theorem 3 shows how to construct in linear time an upward- (upward-rightward-) planar L-drawing from a bitonic (monotonically decreasing) pair. \square

5 Open Problems

Several interesting questions are left open: Can we efficiently test whether a directed plane graph admits a planar L-drawing? Can we efficiently recognize the directed graphs that are edge maximal subject to having a planar L-drawing (they have at most $4n - 6$ edges where n is the number of vertices—see [6]? Does every upward-planar graph have a (not necessarily upward-) planar L-drawing? Can we extend the algorithm for computing a bitonic pair in the variable embedding setting to single-source multi-sink di-graphs? Does every bimodal graph have a planar L-drawing?

References

1. Angelini, P., Da Lozzo, G., Di Bartolomeo, M., Di Donato, V., Patrignani, M., Roselli, V., Tollis, I.G.: L-drawings of directed graphs. In: Freivalds, R.M., Engels, G., Catania, B. (eds.) SOFSEM 2016. LNCS, vol. 9587, pp. 134–147. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49192-8_11
2. Barth, W., Mutzel, P., Yıldız, C.: A new approximation algorithm for bend minimization in the Kandinsky model. In: Kaufmann, M., Wagner, D. (eds.) GD 2006. LNCS, vol. 4372, pp. 343–354. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70904-6_33
3. Bekos, M.A., Kaufmann, M., Krug, R., Siebenhaller, M.: The effect of almost-empty faces on planar Kandinsky drawings. In: Bampis, E. (ed.) SEA 2015. LNCS, vol. 9125, pp. 352–364. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20086-6_27
4. Bläsius, T., Brückner, G., Rutter, I.: Complexity of higher-degree orthogonal graph embedding in the Kandinsky model. In: Schulz, A.S., Wagner, D. (eds.) ESA 2014. LNCS, vol. 8737, pp. 161–172. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44777-2_14
5. Brückner, G.: Higher-degree orthogonal graph drawing with flexibility constraints. Bachelor thesis, Department of Informatics, KIT (2013). https://illwww.iti.kit.edu/_media/teaching/theses/ba-brueckner-13.pdf

6. Chaplick, S., Chimani, M., Cornelsen, S., Da Lozzo, G., Nöllenburg, M., Patrignani, M., Tollis, I.G., Wolff, A.: Planar L-drawings of directed graphs. [arXiv:1708.09107](https://arxiv.org/abs/1708.09107), Cornell University (2017)
7. Di Battista, G., Tamassia, R.: On-line planarity testing. *SIAM J. Comput.* **25**, 956–997 (1996). <https://doi.org/10.1137/S0097539794280736>
8. Di Battista, G., Tamassia, R.: Algorithms for plane representations of acyclic digraphs. *Theor. Comput. Sci.* **61**, 175–198 (1988). [https://doi.org/10.1016/0304-3975\(88\)90123-5](https://doi.org/10.1016/0304-3975(88)90123-5)
9. Di Battista, G., Tamassia, R.: On-line graph algorithms with SPQR-trees. In: Paterson, M.S. (ed.) *ICALP 1990*. LNCS, vol. 443, pp. 598–611. Springer, Heidelberg (1990). <https://doi.org/10.1007/BFb0032061>
10. Didimo, W., Liotta, G., Patrignani, M.: On the complexity of HV-rectilinear planarity testing. In: Duncan, C., Symvonis, A. (eds.) *GD 2014*. LNCS, vol. 8871, pp. 343–354. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45803-7_29
11. Eigelsperger, M.: Automatic layout of UML class diagrams: a topology-shape-metrics approach. Ph.D. thesis, Eberhard-Karls-Universität zu Tübingen (2003)
12. Fößmeier, U., Kaufmann, M.: Drawing high degree graphs with low bend numbers. In: Brandenburg, F.J. (ed.) *GD 1995*. LNCS, vol. 1027, pp. 254–266. Springer, Heidelberg (1996). <https://doi.org/10.1007/BFb0021809>
13. Garg, A., Tamassia, R.: On the computational complexity of upward and rectilinear planarity testing. *SIAM J. Comput.* **31**(2), 601–625 (2001). <https://doi.org/10.1137/S0097539794277123>
14. Gronemann, M.: Bitonic *st*-orderings of biconnected planar graphs. In: Duncan, C., Symvonis, A. (eds.) *GD 2014*. LNCS, vol. 8871, pp. 162–173. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45803-7_14
15. Gronemann, M.: Algorithms for incremental planar graph drawing and two-page book embeddings. Ph.D. thesis, University of Cologne (2015). <http://kups.ub.uni-koeln.de/id/eprint/6329>
16. Gronemann, M.: Bitonic *st*-orderings for upward planar graphs. In: Hu, Y., Nöllenburg, M. (eds.) *GD 2016*. LNCS, vol. 9801, pp. 222–235. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_18. <https://arxiv.org/abs/1608.08578>
17. Kornaropoulos, E.M., Tollis, I.G.: Overloaded orthogonal drawings. In: van Kreveld, M., Speckmann, B. (eds.) *GD 2011*. LNCS, vol. 7034, pp. 242–253. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-25878-7_24

NodeTrix Planarity Testing with Small Clusters

Emilio Di Giacomo¹(✉), Giuseppe Liotta¹, Maurizio Patrignani²,
and Alessandra Tappini¹

¹ Università degli Studi di Perugia, Perugia, Italy
{emilio.digiaco, giuseppe.liotta}@unipg.it,
alessandra.tappini@studenti.unipg.it

² Roma Tre University, Rome, Italy
patrigna@dia.uniroma3.it

Abstract. We study the NodeTrix planarity testing problem for flat clustered graphs when the maximum size of each cluster is bounded by a constant k . We consider both the case when the sides of the matrices to which the edges are incident are fixed and the case when they can be arbitrarily chosen. We show that NodeTrix planarity testing with fixed sides can be solved in $O(k^{3k+\frac{3}{2}}n^3)$ time for every flat clustered graph that can be reduced to a partial 2-tree by collapsing its clusters into single vertices. In the general case, NodeTrix planarity testing with fixed sides can be solved in $O(n^3)$ time for $k = 2$, but it is NP-complete for any $k \geq 3$. NodeTrix planarity testing remains NP-complete also in the free side model when $k > 4$.

1 Introduction

Motivated by the need of visually exploring non-planar graphs, hybrid planarity is one of the emerging topics in graph drawing (see, e.g., [1–3, 9]). A hybrid planar drawing of a non-planar graph suitably represents in restricted geometric regions those dense subgraphs for which a classical node-link representation paradigm would not be visually effective. These regions are connected by edges that do not cross each other. Different representation paradigms for the dense subgraphs give rise to different types of hybrid planar drawings.

Angelini *et al.* [1] consider hybrid planar drawings where dense portions of the graph are represented as intersection graphs of sets of rectangles and study the complexity of testing whether a non-planar graph admits such a representation. In the context of social network analysis, Henry *et al.* [9] introduce NodeTrix representations, where the dense subgraphs are represented as adjacency matrices. Batagelj *et al.* [2] study the question of minimizing the size of the matrices in a NodeTrix representation of a graph while guaranteeing the planarity of the edges that connect different matrices. While Batagelj *et al.* can choose the subgraphs to be represented as matrices, Da Lozzo *et al.* [3] consider the problem of testing whether a flat clustered graph (i.e. a graph with clusters and no sub-clusters) admits a NodeTrix planar representation. In the paper of Da Lozzo *et al.* each cluster must be represented by a different adjacency matrix and the

inter-cluster edges are represented as non-intersecting simple Jordan arcs. They prove that NodeTrix planarity testing for flat clustered graphs is NP-hard even in the constrained case where for each matrix it is specified which inter-cluster edges must be incident on the top, or on the left, or on the bottom, or on the right side.

Motivated by these hardness results, in this paper we study whether NodeTrix planarity testing can be efficiently solved when the size of the clusters is not “too big”. More precisely, we consider flat clustered graphs whose clusters have size bounded by a fixed parameter k and we want to understand whether the NodeTrix planarity testing problem is fixed parameter tractable, i.e. it can be solved in time $f(k)T(n)$, where $T(n)$ is a polynomial in n and $f(k)$ is a function that depends only on k . Our main results can be listed as follows:

- We describe an $O(k^{3k+\frac{3}{2}} \cdot n^3)$ -time algorithm to test NodeTrix planarity with fixed sides for flat clustered graphs that are partial 2-trees. Informally, a flat clustered graph G is a partial 2-tree if the graph obtained by collapsing every cluster of G into a single vertex is a partial 2-tree.
- When the flat clustered graph is not a partial 2-tree, NodeTrix planarity testing with fixed sides can still be solved in $O(n^3)$ time for $k = 2$, but it becomes NP-complete for any larger value of k .
- Finally, we extend the above hardness result to the free sides model and show that NodeTrix planarity testing remains NP-complete when the maximum cluster dimension is larger than four. This is done by proving that NAE3SAT is NP-complete even for triconnected Boolean formulas, which may be a result of independent interest.

Our polynomial-time solution solves a special type of the planarity testing problem where the order of the edges around each vertex is suitably constrained to take into account the fact that a vertex of a matrix M has four copies along the four sides of M . It may be worth recalling that Gutwenger *et al.* [7] have considered an apparently similar problem. Namely, they studied planarity testing where the order of the edges around the vertices may not be arbitrarily permuted. Unfortunately, not only our problem does not fall in any of the cases addressed by Gutwenger *et al.*, but it does not seem solvable by introducing a gadget of polynomial size that models the embedding constraints at each vertex. This characteristic associates NodeTrix planarity testing with other known variants of planarity testing, including clustered planarity, where the use of gadgets of polynomial size has been so far an elusive goal.

The rest of the paper is organized as follows. Preliminary definitions are in Sect. 2. Sections 3 and 4 describe a polynomial time algorithm for clustered 2-trees with bounded cluster-size. In Sect. 5 we show that for general flat clustered graphs and fixed sides NodeTrix planarity testing can be solved in polynomial time for $k = 2$ but it is NP-complete for $k \geq 3$. In the same section we extend this completeness result to NodeTrix planarity testing of flat clustered graphs with free sides. Finally open problems can be found in Sect. 6. For reasons of space, some proofs are sketched or omitted and can be found in [6].

2 Preliminaries

We assume familiarity with basic definitions of graph theory and graph drawing and in particular with the notions of block-cut-vertex tree and of SPQR-tree (see, e.g., [4, 8]).

A *flat clustered graph* $G = (V, E, \mathcal{C})$ is a simple graph with vertex set V , edge set E , and a partition \mathcal{C} of V into sets V_1, \dots, V_h , called *clusters*. An edge $(u, v) \in E$ with $u \in V_i$ and $v \in V_j$ is an *intra-cluster edge* if $i = j$ and it is an *inter-cluster edge* if $i \neq j$.

A *NodeTrix representation* of a flat clustered graph G is such that: **(i)** Each cluster V_i with $|V_i| = 1$ (called *trivial cluster*) is represented as a distinct point in the plane. **(ii)** Each cluster V_i with $|V_i| > 1$ (called *non-trivial cluster*) is represented by a symmetric adjacency matrix M_i (with $|V_i|$ rows and columns), where M_i is drawn in the plane so that its boundary is a square with sides parallel to the coordinate axes. **(iii)** There is no intersection between two distinct matrices or between a point representing a vertex and a matrix. **(iv)** Each intra-cluster edge of a cluster V_i is represented by the adjacency matrix M_i . **(v)** Each inter-cluster edge (u, v) with $u \in V_i$ and $v \in V_j$ is represented by a simple Jordan arc connecting a point on the boundary of matrix M_i with a point on the boundary of matrix M_j , where the point on M_i (on M_j) belongs to the column or to the row of M_i (resp. of M_j) associated with u (resp. with v).

A NodeTrix representation of a flat clustered graph G is *planar* if there is no intersection between any two inter-cluster edges (except possibly at common end-points) nor an intersection between an inter-cluster edge and a matrix. A flat clustered graph is *NodeTrix planar* if it admits a planar NodeTrix representation. Figure 1(a) is an example of a NodeTrix planar representation.

A formal definition of the problem investigated in the paper is as follows. Let $G = (V, E, \mathcal{C})$ be a flat clustered graph with n vertices and let k be the maximum cardinality of a cluster in \mathcal{C} . Clustered graph G is *NodeTrix planar with fixed sides* if it has a NodeTrix planar representation where for each inter-cluster edge, the sides of matrices it attaches to is specified as part of the input; G is *NodeTrix planar with free sides* if the sides of the matrices to which inter-cluster edges attach can be arbitrarily chosen.

Let M_i be the matrix representing cluster V_i in a NodeTrix representation of G ; let v be a vertex of V_i and let (u, v) be an inter-cluster edge. Edge (u, v) can intersect the boundary of M_i in four points $p_{v,T}, p_{v,B}, p_{v,L}$, and $p_{v,R}$ since the row and column that represent v in M_i intersect the four sides of the boundary of M_i . We call such points the *top copy*, *bottom copy*, *left copy*, and *right copy* of v in M_i , respectively.

A *side assignment* for $V_i \in \mathcal{C}$ specifies for each inter-cluster edge whether the edge must attach to the matrix M_i representing V_i in its top, left, right, or bottom side. More precisely, a side assignment is a mapping $\phi_i: \bigcup_{j \neq i} E_{i,j} \rightarrow \{T, B, L, R\}$, where $E_{i,j}$ is the set of inter-cluster edges between the clusters V_i and V_j (V_i and V_j are *adjacent* if $E_{i,j} \neq \emptyset$). A *side assignment* for \mathcal{C} is a set Φ of side assignments for each $V_i \in \mathcal{C}$.

We denote as $G = (V, E, \mathcal{C}, \Phi)$ a flat clustered graph $G = (V, E, \mathcal{C})$ with a given side assignment $\Phi = \{\phi_1, \phi_2, \dots, \phi_{|\mathcal{C}|}\}$. Let Γ be a NodeTrix representation of G such that, for every inter-cluster edge $e = (u, v) \in E$ with $u \in V_i$ and $v \in V_j$, the incidence points of e with the matrices M_i and M_j representing V_i and V_j in Γ are exactly the points $p_{u, \phi_i(e)}$ and $p_{v, \phi_j(e)}$, respectively. We call Γ a NodeTrix representation of G *consistent with Φ* . We say that $G = (V, E, \mathcal{C}, \Phi)$ is *NodeTrix planar* if it admits a NodeTrix planar representation consistent with Φ .

An inter-cluster edge is *heavy* if both its end-vertices belong to non-trivial clusters. It is *light* otherwise. A flat clustered graph is *light* if every inter-cluster edge is light. A 1-*subdivision* of a heavy edge $e = (u, v)$ of a flat clustered graph $G = (V, E, \mathcal{C})$ replaces e with a path $u_0 = u, u_1, u_2 = v$ and defines a new flat clustered graph $G' = (V', E', \mathcal{C}')$, where $V' = V \cup \{u_1\}$, $E' = E/e \cup \{(u_0, u_1), (u_1, u_2)\}$, and $\mathcal{C}' = \mathcal{C} \cup \{u_1\}$. The *light reduction* of G is the flat clustered graph G' obtained by performing a 1-subdivision of every heavy inter-cluster edge of G . A consequence of Theorem 1 in [5] about the edge density of NodeTrix planar graphs, is that the light reduction G' of a NodeTrix planar flat clustered graph G has $O(|V|)$ vertices and $O(|V|)$ inter-cluster edges.

Property 1. A flat clustered graph G is NodeTrix planar if and only if its light reduction G' is NodeTrix planar.

Based on Property 1, in the remainder we shall assume that flat clustered graphs are always light and we call them clustered graphs, for short.

The *frame* of a clustered graph $G = (V, E, \mathcal{C})$ is the graph F obtained by collapsing each cluster $V_i \in \mathcal{C}$, with $|V_i| > 1$, into a single vertex c_i of F , called the *representative vertex of V_i in F* . Let c_i and c_j be the two representative vertices of V_i and V_j in F , respectively. For every inter-cluster edge connecting a vertex of V_i to a vertex of V_j in G there is an edge in F connecting c_i and c_j . Observe that the frame graph F of G is in general a multigraph; however, F is simple when G is light.

Since the NodeTrix planarity of a clustered graph implies the planarity of its frame graph, we will test NodeTrix planarity only on those clustered graphs that have a planar frame.

A *2-tree* is a graph recursively defined as follows: (i) an edge is a 2-tree; (ii) the graph obtained by adding a vertex v to a 2-tree G and by connecting v to two adjacent vertices of G is a 2-tree. A (planar) graph is a *partial 2-tree* if it is a subgraph of a (planar) 2-tree. A biconnected partial 2-tree is a *series-parallel graph*. A clustered graph is a *partial 2-tree* if its frame is a partial 2-tree. We will sometimes talk about series-parallel clustered graphs when their frames are series parallel.

3 NodeTrix Representations and Wheel Reductions

The polynomial-time algorithms described in Sects. 4 and 5 are based on decomposing the planar frame F of a clustered graph $G = (V, E, \mathcal{C}, \Phi)$ into its biconnected components and storing them into a block-cut-vertex tree. We process

each block of F by using an SPQR decomposition tree that is rooted at a reference edge and visited from the leaves to the root. For each visited node μ of the decomposition tree of a block of F we test whether the subgraph of G whose frame is the pertinent graph of μ satisfies the planar constraints imposed by the side assignment on the inter-cluster edges. A key ingredient to efficiently perform the test at μ is the notion of *wheel replacement*.

Let $G = (V, E, \mathcal{C}, \Phi)$ be a clustered graph with side assignment Φ and let $V_i \in \mathcal{C}$ be a cluster with $k > 1$ vertices. V_i admits $k!$ permutations of its vertices and we associate a suitable graph to each such permutation. Let $\pi_i = v_0, v_1, \dots, v_{k-1}$ be a permutation of the vertices of V_i . The *wheel of V_i consistent with π_i* is the wheel graph consisting of a vertex v of degree $4k$ adjacent to the vertices of an oriented cycle $v_{0,T}, v_{1,T}, \dots, v_{k-1,T}, v_{0,R}, v_{1,R}, \dots, v_{k-1,R}, v_{k-1,B}, v_{k-2,B}, \dots, v_{0,B}, v_{k-1,L}, v_{k-2,L}, \dots, v_{0,L}$ where each edge of the cycle is oriented forward. Intuitively, this oriented cycle will be embedded clockwise to encode the constraints induced by a matrix M_i representing V_i when its left-to-right order of columns is π_i . More precisely, a *wheel replacement of cluster V_i consistent with π_i* is the clustered graph obtained as follows: (i) remove V_i and all the inter-cluster edges incident to V_i ; (ii) insert the wheel W_i of V_i consistent with π_i ; and (iii) for each inter-cluster edge $e = (u, v_j)$, with $v_j \in V_i$, insert edge $(u, v_{j, \phi_i(e)})$ incident to W_i . We call edge $(u, v_{j, \phi_i(e)})$ the *image* of edge $e = (u, v_j)$.

Let $G = (V, E, \mathcal{C}, \Phi, \Pi)$ be a clustered graph with side assignment Φ where Π is a set of permutations $\{\pi_1, \pi_2, \dots, \pi_{|\mathcal{C}|}\}$, one for each cluster V_i (with $i = 1, \dots, |\mathcal{C}|$). We call Π the *permutation assignment* of G and we say that G is *NodeTriX planar with side assignment Φ and permutation assignment Π* if G admits a NodeTriX planar representation with side assignment Φ where for each matrix M_i the permutation of its columns is π_i . The *wheel reduction of G consistent with Π* is the graph obtained by performing a wheel replacement of $V_i \in \mathcal{C}$ consistent with π_i for each $i = 1, \dots, |\mathcal{C}|$.

Theorem 1. *Let $G = (V, E, \mathcal{C}, \Phi, \Pi)$ be a clustered graph with side assignment Φ and permutation assignment Π . G is NodeTriX planar if and only if the planar wheel reduction of G admits a planar embedding where the external oriented cycle of each wheel W_i is embedded clockwise.*

Figure 1(a) and (b) show respectively a NodeTriX planar representation of a clustered graph G and the corresponding wheel reduction with its planar embedding.

Based on Theorem 1, we can test the graph $G = (V, E, \mathcal{C}, \Phi)$ for NodeTriX planarity by exploring the space of the possible permutation sets Π and corresponding wheel reductions in search of a NodeTriX planar $G = (V, E, \mathcal{C}, \Phi, \Pi)$. Note that, if the maximum size of a cluster is given as a parameter k , every cluster V_i can be replaced by $k!$ wheel graphs, one for each possible permutation of the vertices of V_i . In order to test planarity, for any such wheel replacement W_i , the cyclic order of the inter-cluster edges incident to the same vertex of W_i can be arbitrarily permuted. While each wheel reduction yields an instance of constrained planarity testing that can be solved with the linear-time algorithm

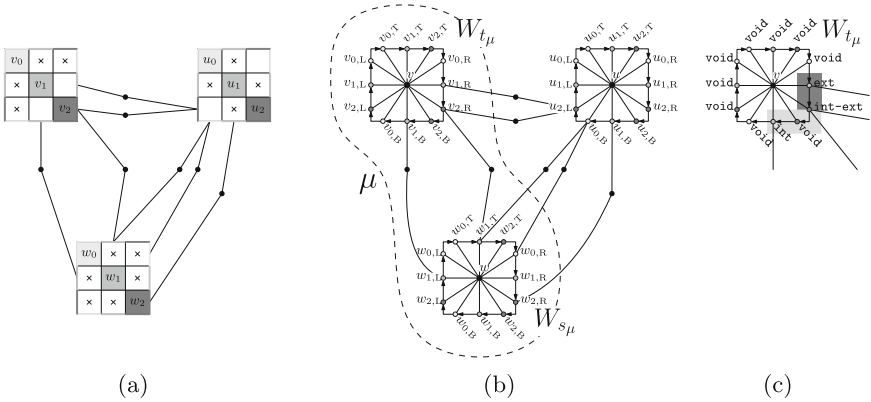


Fig. 1. (a) A NodeTrix planar representation of a clustered graph. (b) The planar embedding of the corresponding wheel reduction. (c) Labeling of the vertices of W_{t_μ} ; the complete internal and external sequences are highlighted.

described in [7], a brute-force approach that repeats this algorithm on each possible wheel reduction may lead to testing planarity on $|\mathcal{C}|^{k!}$ different instances. Instead, for each visited node μ of the decomposition tree T we compute a succinct description of the possible NodeTrix planar representations of the subgraph G_μ of G represented by the subtree of T rooted at μ . This is done by storing for the poles of μ those pairs of wheel graphs that are compatible with a NodeTrix planar representation of G_μ . How to efficiently compute such a succinct description will be the subject of the next sections.

4 Testing NodeTrix Planarity for Partial 2-Trees

In this section we prove that NodeTrix planarity testing with fixed sides can be solved in polynomial time for a clustered graph $G = (V, E, \mathcal{C}, \Phi)$ when the maximum size of any cluster of \mathcal{C} is bounded by a constant and the frame graph is a partial 2-tree. This contrasts with the NP-hardness of NodeTrix planarity testing with fixed sides proved in [3] in the case where the size of the clusters is unbounded.

We first study the case of a clustered graph whose frame graph is a series-parallel graph, i.e., it is biconnected and its SPQR decomposition tree only has Q-, P-, and S-nodes. We then consider the case of partial 2-trees, i.e., graphs whose biconnected components are series-parallel.

4.1 Series-Parallel Frame Graphs

In this section we prove that NodeTrix planarity testing with fixed sides can be solved in $O(k^{3k+\frac{3}{2}} \cdot n^2)$ time for clustered graphs whose frame graphs are series-parallel and have cluster size at most k .

Let $G = (V, E, \mathcal{C}, \Phi)$ be a series-parallel clustered graph with side assignment Φ and let F be its frame graph. Let T be the SPQ decomposition tree of F rooted at any Q-node. To simplify the description and without loss of generality, we assume that every S-node of T has exactly two children. Let μ be a node of T , and let s_μ, t_μ be the poles of μ . Consider the pertinent graph F_μ represented by the subtree of T rooted at μ and let v_μ be a pole of μ ($v_\mu \in \{s_\mu, t_\mu\}$). Pole v_μ in the frame graph F may correspond to a non-trivial cluster V_i of \mathcal{C} . In this case, we call v_μ a *non-trivial pole of μ* and cluster V_i the *pertinent cluster of v_μ* .

The edges of F_μ incident to v_μ are the *intra-component edges of v_μ* . The other edges of F incident to v_μ are the *extra-component edges of v_μ* . Each intra-component (extra-component) edge of v_μ corresponds to an inter-cluster edge e' of G incident to one vertex of the pertinent cluster V_μ of v_μ . We call e' an *intra-component edge (extra-component edge) of V_μ* . We associate $k!$ wheel graphs to each non-trivial pole v_μ of μ . Each of them is a wheel replacement of the pertinent cluster of v_μ , consistent with one of the $k!$ permutations of its vertices.

Let v_μ be a non-trivial pole of μ , let V_μ be the pertinent cluster of v_μ , let π_μ be a permutation of the vertices of V_μ , and let W_μ be the wheel replacement of V_μ consistent with π_μ . Every edge e incident to W_μ such that e is the image of an inter-cluster edge e' of G is labeled either **int** or **ext**, depending on whether e' is an intra-component or an extra-component edge of V_μ . A vertex w of the external cycle of W_μ is assigned one label of the set $\{\text{void}, \text{int}, \text{ext}, \text{int-ext}\}$ as follows. Vertex w is labeled **void** if no edge incident to w is the image of an inter-cluster edge. Vertex w is labeled **int** (resp. **ext**) if we have a label **int** (resp. **ext**) on every edge e incident to w such that e is the image of an inter-cluster edge. Otherwise, vertex w is labeled **int-ext**. See Fig. 1(c) for an example concerning the wheel W_{t_μ} of Fig. 1(b); the dashed curve of Fig. 1(b) shows the subgraph of the wheel reduction corresponding to F_μ .

A clockwise sequence v_0, v_1, \dots, v_j of vertices of the external cycle of W_μ is an *external sequence of pole v_μ consistent with π_μ* if v_0 and v_j are labeled either **ext** or **int-ext** and all the other vertices of the sequence are labeled either **void** or **ext**. An external clockwise sequence of pole v_μ is *complete* if it contains all the vertices of W_μ that are labeled **ext** and **int-ext**. Note that a complete external sequence may contain many **void** vertices but no **int** vertex. *Internal* and *complete internal* sequences of pole v_μ are defined analogously. Observe that a complete internal sequence and a complete external sequence of v_μ may not exist when vertices labeled **int** and vertices labeled **ext** alternate more than twice when traversing clockwise the external cycle of W_μ , or when three vertices are labeled **int-ext**. A special case is when W_μ has exactly two vertices w_1 and w_2 labeled **int-ext** and all other vertices are **void**. In this case, the clockwise sequence from w_1 to w_2 and the clockwise sequence from w_2 to w_1 are both complete internal and complete external sequences.

In order to test $G = (V, E, \mathcal{C}, \Phi)$ for NodeTriX planarity, we implicitly take into account all possible permutation assignments Π by considering, for each non-trivial pole w_μ of each node μ of T , its $k!$ possible wheels and by computing their complete internal and complete external sequences. We visit the *SPQ*

decomposition tree T from the leaves to the root and equip each node μ of T with information regarding the complete internal and complete external sequences of its non-trivial poles. Let μ be an internal node of T , let v_μ be a non-trivial pole of μ , let π_{v_μ} be a permutation of the pertinent cluster V_μ of v_μ , and let W_μ be the wheel of V_μ consistent with π_{v_μ} . We denote as $ISeq(\mu, v_\mu, \pi_{v_\mu})$ the complete internal sequence of v_μ consistent with π_{v_μ} in pole μ and as $ESeq(\mu, v_\mu, \pi_{v_\mu})$ the complete external sequence of v_μ consistent with π_{v_μ} in pole μ . We distinguish between the different types of nodes of T .

Node μ is a Q-node. Since G is light, at most one of its poles is non-trivial. Let e be an edge of F that is the pertinent graph of μ . One end-vertex of e is the representative vertex in F of the pertinent cluster of the non-trivial pole v_μ . In fact, edge e corresponds to an edge $e' = (u, z)$ of G such that $u \in V_\mu$ and z is a trivial cluster. The side assignment ϕ_{v_μ} defines whether e is incident to the top, bottom, left, or right copy u_W of u in the wheel W_μ of V_μ . For any possible permutation π_{v_μ} we have $ISeq(\mu, v_\mu, \pi_{v_\mu}) = u_W$. If u_W is labeled **int-ext**, then $ESeq(\mu, v_\mu, \pi_{v_\mu})$ is the external cycle of W_μ starting at u_W and ending at u_W . Otherwise, traverse the external cycle of W_μ starting at u_W and following the direction of the edges; $ESeq(\mu, v_\mu, \pi_{v_\mu})$ consists of all the encountered vertices from the first labeled **ext** to the last labeled **ext**.

Node μ is a P-node. Let $\nu_0, \nu_1, \dots, \nu_{h-1}$ be the children of μ . Observe that v_μ is a non-trivial pole also for the children $\nu_0, \nu_1, \dots, \nu_{h-1}$ of μ . We consider every permutation π_{v_μ} such that $\nu_0, \nu_1, \dots, \nu_{h-1}$ have both a complete internal sequence and a complete external sequence compatible with π_{v_μ} . The complete internal sequence of v_μ consistent with π_{v_μ} is the union of the complete internal sequences of the children $\nu_0, \nu_1, \dots, \nu_{h-1}$, that is $ISeq(\mu, v_\mu, \pi_{v_\mu}) = \cup_{i=0}^{h-1} ISeq(\nu_i, v_\mu, \pi_{v_\mu})$.

To determine the complete external sequence of v_μ consistent with π_{v_μ} we consider the intersection of the complete external sequences of the children of μ . If this intersection consists of exactly one sequence of consecutive vertices, then $ESeq(\mu, v_\mu, \pi_{v_\mu}) = \cap_{i=0}^{h-1} ESeq(\nu_i, v_\mu, \pi_{v_\mu})$. Otherwise (i.e., the intersection is empty or it consists of more than one sequence of consecutive vertices), v_μ does not have a complete external sequence consistent with π_{v_μ} .

Node μ is an S-node. Let ν be the child of μ that shares the pole v_μ with μ . We consider every permutation π_{v_μ} such that ν has both $ISeq(\nu, v_\mu, \pi_{v_\mu})$ and $ESeq(\nu, v_\mu, \pi_{v_\mu})$. The complete internal (external) sequence of v_μ consistent with π_{v_μ} is $ISeq(\mu, v_\mu, \pi_{v_\mu}) = ISeq(\nu, v_\mu, \pi_{v_\mu})$ ($ESeq(\mu, v_\mu, \pi_{v_\mu}) = ESeq(\nu, v_\mu, \pi_{v_\mu})$).

To test G for NodeTrix planarity we execute a bottom-up traversal of T and, for each node μ with poles s_μ and t_μ , we check whether each possible pair $(\pi_{s_\mu}, \pi_{t_\mu})$ induces complete internal and external sequences for s_μ and t_μ that are ‘compatible’ with a planar embedding of the wheel reduction of G . If this is the case, by Theorem 1, G is NodeTrix planar, otherwise we reject G .

More formally, let π_{s_μ} (π_{t_μ} , respectively) be a permutation such that s_μ (t_μ , respectively) has both a complete internal sequence and a complete external sequence compatible with π_{s_μ} (π_{t_μ} , respectively). We say that $(\pi_{s_\mu}, \pi_{t_\mu})$ is a

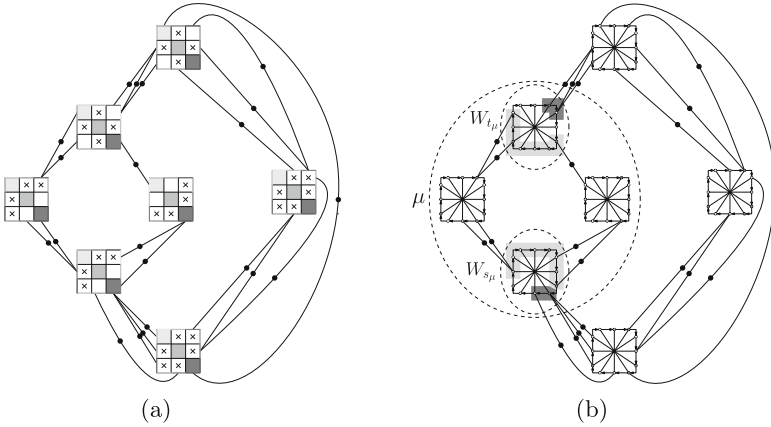


Fig. 2. (a) A NodeTrix planar representation Γ of $G = (V, E, \mathcal{C}, \Phi)$. (b) Γ induces a permutation assignment and a planar embedding of a wheel reduction of G ; the complete internal and external sequences for a pair of poles are also highlighted.

compatible pair of permutations for μ if either one of the poles is a trivial pole or one of the following cases applies.

Node μ is a Q-node. In this case all $k!$ possible pairs of permutations for s_μ or t_μ (recall that only one of them is non-trivial) are compatible for μ .

Node μ is a P-node. Let $\nu_0, \nu_1, \dots, \nu_{h-1}$ be the children of μ . Consider a pair of permutations $(\pi_{s_\mu}, \pi_{t_\mu})$; we recall that, for $i = 0, \dots, h-1$, each ν_i has poles s_μ and t_μ . A first condition for pair $(\pi_{s_\mu}, \pi_{t_\mu})$ to be a compatible pair for μ is that $(\pi_{s_\mu}, \pi_{t_\mu})$ is also a compatible pair for ν_i , with $i = 0, \dots, h-1$. A second condition asks that the pair $(\pi_{s_\mu}, \pi_{t_\mu})$ defines opposite orders on the poles of μ . Namely, let W_μ^s (resp., W_μ^t) be the wheel of V_{s_μ} (resp., V_{t_μ}) consistent with π_{s_μ} (resp., π_{t_μ}). Traversing clockwise the external cycle of W_μ^s starting from the first vertex of $ESeq(\mu, s_\mu, \pi_{s_\mu})$, let $ISeq(\nu_0, s_\mu, \pi_{s_\mu}), ISeq(\nu_1, s_\mu, \pi_{s_\mu}), \dots, ISeq(\nu_{h-1}, s_\mu, \pi_{s_\mu})$ be the order by which the internal sequences are encountered. Pair $(\pi_{s_\mu}, \pi_{t_\mu})$ defines opposite orders on the poles of μ if, traversing clockwise the external cycle of W_μ^t starting from the first vertex of $ESeq(\mu, t_\mu, \pi_{t_\mu})$, the order by which we encounter the internal sequences of $\nu_0, \nu_1, \dots, \nu_{h-1}$ is the opposite one, i.e., the order is $ISeq(\nu_{h-1}, t_\mu, \pi_{t_\mu}), ISeq(\nu_{h-2}, t_\mu, \pi_{t_\mu}), \dots, ISeq(\nu_0, t_\mu, \pi_{t_\mu})$.

Node μ is an S-node. Let ν_0 and ν_1 be the children of μ such that $s_{\nu_0} = s_\mu$, $t_{\nu_0} = s_{\nu_1}$, and $t_{\nu_1} = t_\mu$. A pair $(\pi_{s_\mu}, \pi_{t_\mu})$ is a compatible pair for μ if there exists a permutation $\pi_{t_{\nu_0}}$ such that the pair $(\pi_{s_\mu}, \pi_{t_{\nu_0}})$ is compatible for ν_0 and the pair $(\pi_{t_{\nu_0}}, \pi_{t_\mu})$ is compatible for ν_1 .

Figure 2 suggests that a NodeTrix planar representation of a clustered graph G defines a permutation assignment Π such that, for every node μ of T , pair $(\pi_{s_\mu}, \pi_{t_\mu})$ is a compatible pair for μ .

Lemma 1. *Let $G = (V, E, \mathcal{C}, \Phi)$ be a clustered graph with side assignment Φ and let T be the SPQ decomposition tree of the frame graph of G . Graph G is NodeTrix planar if and only if there exists a permutation assignment Π such that, for every node μ of T with poles s_μ and t_μ , we have that permutation $\pi_{s_\mu} \in \Pi$ and permutation $\pi_{t_\mu} \in \Pi$ form a compatible pair of permutations for μ .*

Lemma 2. *Let $G = (V, E, \mathcal{C}, \Phi)$ be a series-parallel clustered graph with side assignment Φ . Let k be the maximum size of any cluster in \mathcal{C} and let n be the cardinality of V . There exists an $O(k^{3k+\frac{3}{2}} \cdot n^2)$ -time algorithm that tests whether G is NodeTrix planar with side assignment Φ and if so, it computes a NodeTrix planar representation of G consistent with Φ .*

Proof. Let F be the frame graph of G ; for any possible choice of an edge e of F we repeat the following procedure. We construct the SPQ decomposition tree of G rooted at the Q-node whose pertinent graph is e . We visit T from the leaves to the root and test whether G has a permutation assignment Π such that $G = (V, E, \mathcal{C}, \Phi, \Pi)$ is NodeTrix planar. We first equip each non-trivial pole v_μ of every node μ of T with its possible complete internal and complete external sequences. The maximum number of complete internal sequences of v_μ is $k!$. The same is true for the complete external sequences. If each complete (internal or external) sequence of pole v_μ is encoded by means of its first and last vertex in the clockwise order around W_{v_μ} , then each complete internal or external sequence needs constant space. It follows that the intersection or the union of two complete internal or external sequences can be computed in constant time. Therefore, all complete internal and external sequences for each non-trivial pole of T can be computed in $O(k!)$ time. Hence, the whole bottom-up traversal to equip all non-trivial poles with every possible complete internal/external sequence can be executed in $O(k! \cdot n)$ time. We now test whether there exists a permutation assignment Π such that any node μ of T has a compatible pair of permutations. To this aim, we look at the complete internal and external sequences for the pair of poles of the children of μ . For each pair $(\pi_{s_\mu}, \pi_{t_\mu})$ of permutations of the poles of μ we equip μ with the information about whether such pair is compatible for μ . This requires $O(k!^2)$ space. If μ is a Q-node, every pair of permutations $(\pi_{s_\mu}, \pi_{t_\mu})$ is compatible for μ . It follows that all compatible pairs for μ can be computed in $O(k!)$ time (recall that one between s_μ and t_μ is non-trivial) and, hence, in $O(k! \cdot n)$ time for all the Q-nodes of T . If μ is a P-node with children $\nu_0, \nu_1, \dots, \nu_{h-1}$, π_{s_μ} is one of the permutations that equip s_μ , and π_{t_μ} is one of the permutations that equip t_μ , testing whether the pair $(\pi_{s_\mu}, \pi_{t_\mu})$ is a compatible pair for μ can be executed in $O(h)$ time. It follows that all compatible pairs for μ can be computed in $O(k!^2 \cdot h)$ time and, hence, in $O(k!^2 \cdot n)$ time for all P-nodes of T . If μ is an S-node with children ν_0 and ν_1 , π_{s_μ} is one of the permutations that equip s_μ , and π_{t_μ} is one of the permutations that equip t_μ , testing whether the pair $(\pi_{s_\mu}, \pi_{t_\mu})$ is a compatible pair for μ can be executed in $O(k!)$ time, corresponding to choosing all possible permutations for the pole shared between ν_0 and ν_1 . It follows that all compatible pairs for μ can be computed in $O(k!^3)$ time and, hence, in $O(k!^3 \cdot n)$ time for all S-nodes of T .

In conclusion, the time complexity of a bottom-up visit of T rooted at e is $O(k!^3 \cdot n)$. By rooting T at all possible Q-nodes, we obtain an overall time complexity of $O(k!^3 \cdot n^2)$. By Stirling's approximation, $k! \sim \sqrt{2\pi k} (\frac{k}{e})^k$ and thus a series-parallel clustered graph G with n vertices, side assignment Φ , and maximum cluster size k can be tested for NodeTriX planarity in $O(k^{3k+\frac{3}{2}} \cdot n^2)$ time. Note that the compatible pair of permutations stored at each node μ of T implicitly define a planar embedding of a wheel reduction of G . It can be shown that it is possible to construct a NodeTriX planar representation of G in time proportional to the number of edges of G , which is $O(n \cdot k)$ [5]. The statement of the lemma follows. \square

4.2 Partial 2-Trees

We now consider clustered graphs whose cluster size is at most k and such that their frame graph is a partial 2-tree, i.e. it is a planar graph whose biconnected components are series-parallel. We handle this case by decomposing the frame graph into its blocks and we store them into a block-cut-vertex tree. The following theorem generalizes the result of Lemma 2.

Theorem 2. *Let $G = (V, E, \mathcal{C}, \Phi)$ be a partial 2-tree clustered graph with side assignment Φ . Let k be the maximum size of any cluster in \mathcal{C} and let n be the cardinality of V . There exists an $O(k^{3k+\frac{3}{2}} \cdot n^3)$ -time algorithm that tests whether G is NodeTriX planar with side assignment Φ and if so, it computes a NodeTriX planar representation of G consistent with Φ .*

5 General Planar Frame Graphs

In this section we study the problem of extending Theorem 2 to planar frame graphs that may not be partial 2-trees. We prove that NodeTriX planarity testing with fixed sides can be solved in polynomial time for maximum cluster size $k = 2$. However, the problem becomes NP-complete with fixed sides for $k \geq 3$ and it remains NP-complete even in the free sides scenario for $k \geq 5$.

Every block of the frame graph can be decomposed into its triconnected components by means of an SPQR decomposition tree. For each block, we adopt the same approach as for series-parallel graphs and look for a permutation assignment Π such that, for every pair of poles s_μ and t_μ , $(\pi_{s_\mu}, \pi_{t_\mu})$ forms a compatible pair for μ when μ is either a Q-node, a P-node, or an S-node. We extend the definition of compatible pairs of permutations for an R-node as follows.

Let $G = (V, E, \mathcal{C}, \Phi)$ be a clustered graph with side assignment Φ , let F be the frame graph of G , and let T be the SPQR decomposition tree of F . Let μ be an R-node of T with poles s_μ and t_μ . A pair of permutations $(\pi_{s_\mu}, \pi_{t_\mu})$ forms a *compatible pair for μ* if there exists a planar embedding of the skeleton $\text{skel}(\mu)$ of μ for which the following conditions hold: (i) For each vertex v of $\text{skel}(\mu)$, let e_0, e_1, \dots, e_{h-1} be the virtual edges of $\text{skel}(\mu)$ incident to v in clockwise order around v . Each such edge e_i is associated with a child ν_i of μ . There exists

a permutation π_v such that the complete internal sequences $ISeq(\nu_0, v, \pi_v)$, $ISeq(\nu_1, v, \pi_v), \dots, ISeq(\nu_{h-1}, v, \pi_v)$ appear in this clockwise order around v . (ii) Every vertex v of $skel(\mu)$ can be assigned a permutation π_v such that: $\pi_v = \pi_{s_\mu}$ if $v = s_\mu$ and $\pi_v = \pi_{t_\mu}$ if $v = t_\mu$, and for each virtual edge $e = (u, v)$ in $skel(\mu)$ that corresponds to a child ν of μ , the permutation pair (π_u, π_v) is compatible for ν .

Observe that, in the case of maximum cluster size $k = 2$, the possible permutations of the induced cluster V_v of a vertex v of $skel(\mu)$ are exactly two, denoted by π_v^+ and π_v^- . In order to test whether $(\pi_{s_\mu}, \pi_{t_\mu})$ forms a compatible pair for μ , we perform a traversal of $skel(\mu)$ starting at s_μ . Permutation π_{s_μ} and the clockwise order of the edges incident to s_μ can impose to choose only one of the two permutations $\pi_w^+ \circ \pi_w^-$ available for each vertex w adjacent to s_μ and corresponding to a non-trivial cluster of G . Each such w and its incident edges, in turn, propagate constraints on the possible permutations to their neighbors, till t_μ is reached. Therefore, testing whether π_{s_μ} and π_{t_μ} form a compatible pair for μ can be reduced to a suitable problem of labeling the edges and vertices of $skel(\mu)$ and verifying that at the end s_μ and t_μ are labeled with π_{t_μ} and π_{s_μ} .

Theorem 3. *Let $G = (V, E, \mathcal{C}, \Phi)$ be an n -vertex clustered graph with side assignment Φ such that the maximum size of any cluster in \mathcal{C} is two. There exists an $O(n^3)$ -time algorithm that tests whether G is NodeTrix planar with the given side assignment and if so, computes a NodeTrix planar representation of G consistent with Φ .*

The proof of the following theorem is based on a reduction from (non-planar) NAE3SAT.

Theorem 4. *NodeTrix planarity testing with fixed sides and cluster size at most k is NP-complete for any $k \geq 3$.*

Now, we extend the above hardness result to the free sides model and show that NodeTrix planarity testing remains NP-complete when the maximum cluster dimension is larger than four. This is done by proving that NAE3SAT is NP-complete even for triconnected Boolean formulas, which may be a result of independent interest.

Theorem 5. *NAE3SAT is NP-complete for triconnected Boolean formulas.*

Theorem 6. *NodeTrix planarity testing with free sides and cluster size at most k is NP-complete for any $k \geq 5$.*

6 Open Problems

We conclude the paper by listing some open problems that, in our opinion, are worth investigating. (i) Study the complexity of NodeTrix planarity testing in the free sides scenario for values of k between 2 and 5. (ii) Study families of clustered graphs for which NodeTrix planarity testing is fixed parameter tractable in the free sides scenario. (iii) Determine whether the time complexity of the algorithms in Theorems 2 and 3 can be improved.

References

1. Angelini, P., Da Lozzo, G., Di Battista, G., Frati, F., Patrignani, M., Rutter, I.: Intersection-link representations of graphs. *J. Graph Algorithms Appl.* **21**(4), 731–755 (2017)
2. Batagelj, V., Brandenburg, F., Didimo, W., Liotta, G., Palladino, P., Patrignani, M.: Visual analysis of large graphs using (X, Y)-clustering and hybrid visualizations. *IEEE Trans. Vis. Comput. Graph.* **17**(11), 1587–1598 (2011)
3. Da Lozzo, G., Di Battista, G., Frati, F., Patrignani, M.: Computing NodeTriX representations of clustered graphs. In: Hu, Y., Nöllenburg, M. (eds.) GD 2016. LNCS, vol. 9801, pp. 107–120. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_9
4. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: *Graph Drawing*. Prentice Hall, Upper Saddle River (1999)
5. Di Giacomo, E., Liotta, G., Patrignani, M., Tappini, A.: NodeTriX Planarity Testing with Small Clusters. In: Frati, F., Ma, K. (eds.) GD 2017. LNCS, vol. 10692, pp. 479–491. Springer, Cham (2018)
6. Di Giacomo, E., Liotta, G., Patrignani, M., Tappini, A.: NodeTriX planarity testing with small clusters. CoRR 1708.09281 (2017). <http://arxiv.org/abs/1708.09281>
7. Gutwenger, C., Klein, K., Mutzel, P.: Planarity testing and optimal edge insertion with embedding constraints. *J. Graph Algorithms Appl.* **12**(1), 73–95 (2008)
8. Harary, F.: *Graph Theory*. Addison-Wesley Series in Mathematics. Addison Wesley, Reading (1969)
9. Henry, N., Fekete, J., McGuffin, M.J.: NodeTriX: a hybrid visualization of social networks. *IEEE Trans. Vis. Comput. Graph.* **13**(6), 1302–1309 (2007)

The Painter's Problem: Covering a Grid with Colored Connected Polygons

Arthur van Goethem¹, Irina Kostitsyna¹, Marc van Kreveld²,
Wouter Meulemans¹, Max Sondag¹, and Jules Wulms¹✉

¹ Department of Mathematics and Computer Science, TU Eindhoven,
Eindhoven, The Netherlands

{a.i.v.goethem,i.kostitsyna,w.meulemans,m.f.m.sondag,
j.j.h.m.wulms}@tue.nl

² Department of Information and Computing Sciences, Utrecht University,
Utrecht, The Netherlands

m.j.vankreveld@uu.nl

Abstract. Motivated by a new way of visualizing hypergraphs, we study the following problem. Consider a rectangular grid and a set of colors χ . Each cell s in the grid is assigned a subset of colors $\chi_s \subseteq \chi$ and should be partitioned such that for each color $c \in \chi_s$ at least one piece in the cell is identified with c . Cells assigned the empty color set remain white. We focus on the case where $\chi = \{\text{red, blue}\}$. Is it possible to partition each cell in the grid such that the unions of the resulting red and blue pieces form two connected polygons? We analyze the combinatorial properties and derive a necessary and sufficient condition for such a *painting*. We show that if a painting exists, there exists a painting with bounded complexity per cell. This painting has at most five colored pieces per cell if the grid contains white cells, and at most two colored pieces per cell if it does not.

1 Introduction

Hypergraphs are a powerful structure to represent unordered set systems. In general, there are a number of elements (vertices of the hypergraph) and a number of different subsets over these elements (the hyperedges of the graph). The purpose of visualizing hypergraphs is to clarify the various set relations between the hyperedges. There are, roughly speaking, two strands of hypergraph visualizations: those where the position of the elements is fixed (e.g. [2, 7, 8, 15]), and those where the positions can be chosen by the layout algorithm (e.g. [10, 18, 19]). For a more detailed overview and in-depth classification of set visualization methods we refer to the survey by Alsallakh et al. [4]. Though some methods aim to

This work was initiated at the 2nd Workshop on Applied Geometric Algorithms (AGA 2017) supported by the Netherlands Organisation for Scientific Research (NWO), 639.023.208. AvG is supported by NWO 612.001.102; IK by FRS-FNRS; MvK by NWO 612.001.651; WM and JW by NLeSC 027.015.G02; MS by NWO 639.023.208.

overcome layout complexity by replicating elements (e.g. [3,10]), we focus on a visualization using a single representation for each element.

In theoretic research on drawing hypergraphs (e.g. [6,12]), the (often implicit) assumption is that the representations of two sets may cross at common vertices. Such crossings are not deemed problematic as most visual encodings rely on the local *nesting* of intersecting polygons (in line with the prototypical Venn and Euler diagrams [5] and similar visual overlays [7,8,15]) to identify set memberships. Nesting, however, gives a strong visual cue of containment and may result in misleading visual representations implying containment relationships between hyperedges. A rendering style without nesting is one suggested for Kelp Diagrams [8]. However, its cluttered appearance caused it not to feature in the later extension, KelpFusion [15].

One of the most well-established quality criteria of graph drawings is planarity (see e.g. [16,17]). When nested encodings are used, a planar drawing relates to finding a planar support [6]: a planar (regular) graph such that the vertices of each hyperedge induce a connected subgraph in the support. Deciding whether a planar support exists is possible for some simple support classes (see [6] for a discussion), but is already NP-hard for 2-outerplanar support graphs [6]. Optimizing hypergraph supports for total graph length without planarity constraints is NP-hard, but approximation algorithms exist [1,11].

Representations that do not require nesting are edge-based drawings [13] or the equivalent Zykov representation [22], for which notions of planarity follow readily from the standard notion for regular graphs.

Instead we suggest a visual design that uses *disjoint polygons* to present hyperedges: vertices are represented as simple geometric primitives (e.g. a square or circle); hyperedges are represented as connected polygons that overlaps only and all its incident vertices; and all such polygons are pairwise disjoint. As illustrated in Fig. 1, our disjoint-polygons encoding is stronger as it can visualize some hypergraphs that are not Zykov-planar, whereas any Zykov-planar hypergraph admits a disjoint-polygons representation. We can use vertices to “pass in between” the representations of other hyperedges, though not as flexibly as is allowed for planar supports: the polygons must remain disjoint.

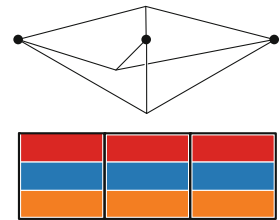


Fig. 1. A hypergraph that is not Zykov-planar (top) but has a disjoint-polygons drawing (bottom).

Contributions. We investigate the properties of drawing hypergraphs using disjoint polygons. Motivated by moving towards a set visualization in a geographic small multiples or grid map (see e.g. [14,23]), we specifically study the variant where each element has a fixed location, being a cell in a rectangular grid. As an initial exploration we focus on the 2-color case, where each cell is either red, blue, both (purple), or uncolored (white). We thus aim to partition each purple cell into red and blue pieces, such that the resulting pieces of a single color form a connected polygon. We derive a necessary and sufficient condition to efficiently

recognize whether an instance is solvable. For solvable instances, we bound the number of colored pieces within each cell by a small constant and show that these bounds are tight. Due to space constraints, some proofs have been shortened or omitted; for full proofs, please refer to the ArXiv version [21].

2 Preliminaries

We define a *k-colored grid* Γ as a rectangular grid, in which each cell s has a set of associated colors $\chi_s \subseteq \{1, \dots, k\}$. A *fully k-colored grid* is the case where $\chi_s \neq \emptyset$ for all cells s . Throughout this paper, we primarily investigate 2-colored grids and use *colored grid* to refer to the 2-colored case, unless indicated otherwise. We refer to the two colors as (*r*)*ed* and (*b*)*lue*; cells for which $\chi_s = \{r, b\}$ are called (*p*)*urple*. Cells with no associated colors are *white*.

A region is a maximal set of cells that have the same color assignment (*r*, *b*, or *p*) and where every cell s in the region is connected via adjacent cells to every other cell s' in the region. Cells are considered adjacent if they are horizontally or vertically adjacent.

A *panel* π_s for cell s (with $\chi_s \neq \emptyset$) maps each color $c \in \chi_s$ to a (possibly disconnected) area $\pi_s(c)$ such that these partition the cell: that is, $\bigcup_{c \in \chi_s} \pi_s(c) = s$ and $\pi_s(c_1) \cap \pi_s(c_2) = \emptyset$ for colors $c_1 \neq c_2$. A *painting* Π of a *k-colored grid* consists of panels π_s for each cell s with $\pi_s(c) \neq \emptyset$ for each $c \in \chi_s$ and $\pi_s(c) = \emptyset$ otherwise. We call a painting *connected* if each color forms a connected polygon: that is, $\bigcup_{s \in \Gamma} \pi_s(c)$ is a connected polygon for each color $c \in \{1, \dots, k\}$. For this definition, two cells sharing only a corner are *not* considered connected. Our primary interest is in connected paintings: in the remainder, we use painting to indicate a connected painting.

3 Characterizing Colored Grids with a Painting

In this section we show how to test whether a 2-colored grid admits a painting and how to find a painting if one exists. As all completely red, blue, and white panels are fixed, finding a painting reduces to finding partitions of purple cells that ensure that the resulting red and blue polygon are connected. We show that this connectivity is of key importance: if we can find suitable connections though the purple regions, then we can also create a partition that results in a valid panel for each cell in the purple regions.

We capture the connectivity options for the red and blue polygon using two embedded graphs, G_r and G_b . We construct these graphs in three steps:

1. Connect red (blue) regions that are adjacent along a purple region's boundary.
2. Remove holes from the purple regions by inserting connections (Sect. 3.3).
3. Construct G_r and G_b using a gadget for purple regions (Sects. 3.1 and 3.2).

For the first step, observe that consecutive (not necessarily distinct) adjacent regions of the same color can always be safely connected via the purple region's

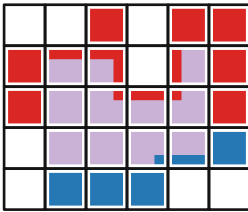


Fig. 2. Safe connections between adjacent same-color regions. (Color figure online)

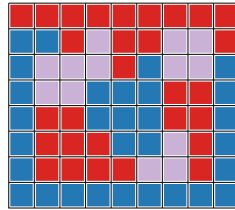
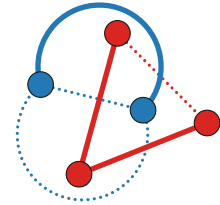


Fig. 3. 2-colored grid with 4 regions around each purple region and corresponding graphs G_r and G_b . (Color figure online)



boundary without restricting the connectivity options for the other color (see Fig. 2). After the first two steps, we represent the remaining red (blue) regions as vertices in G_r (G_b). Edges in G_r and G_b represent connection options through purple regions; intersections indicate a choice to connect either blue or red regions through (part of) a purple region. The gadget for purple regions with many adjacent red and blue regions also requires some additional vertices in these graphs. We prove that these graphs admit a simple characterization of 2-colored grids that admit a painting, as captured in the theorem below.

Theorem 1. *A 2-colored grid Γ admits a painting if and only if the corresponding graphs G_r and G_b are each other’s exact duals: there is exactly one blue vertex in every red face and there is exactly one red vertex in every blue face.*

For explanatory reasons we start with the simplest case: purple regions with at most four neighbors and without holes (Sect. 3.1). Subsequently, we alleviate the assumption on the number of neighbors (Sect. 3.2) and permit holes in the purple regions, by showing how to perform Step 2 (Sect. 3.3).

3.1 Simple Purple Regions

We assume that Step 1 has been performed and a purple region has no holes and at most four adjacent regions. The adjacent red and blue regions of a purple region P form an ordered cyclic list as they appear along the boundary of P and alternate in color (due to Step 1). Let $\kappa(P)$ denote the length of this list for P . κ is even due to color alternation, and by assumption here $\kappa(P) \leq 4$. There can be duplicates in this list as the same red or blue region can touch P multiple times.

Every purple region with $\kappa(P) = 2$ can be painted by creating a spanning tree on the centers of the panels of P in one color and connecting it to the corresponding region. The rest of the panels is colored in the other color. We assume these are handled; what remains is to deal with the regions with $\kappa(P) = 4$.

For a purple region P with $\kappa(P) = 4$, we create a red edge in G_r and a blue edge in G_b that intersect: the red edge connects the red vertices corresponding to

the adjacent red regions; the blue edge connects the corresponding blue vertices. There may be multiple edges between two vertices (see Fig. 3). If the same red or blue region touches the purple region twice, the edge is a self-loop. Every red or blue edge intersects exactly one blue or red edge respectively, and G_r and G_b are plane by construction. Using the following lemma we prove the exact characterization of graphs G_r and G_b of a 2-colored grid Γ that admits a painting.

Lemma 1 ([9, 20]). *Let G be a plane graph, G^* its dual and T a spanning tree of G . Then $T^* = \{e^* \mid e \notin T\}$ is a spanning tree of G^* .*

Lemma 2. *A 2-colored grid Γ in which each purple region P has no holes and $\kappa(P) \leq 4$, admits a painting if and only if the corresponding graphs G_r and G_b are each other's exact duals.*

Proof (sketch). We prove that if Γ admits a painting then graphs G_r and G_b are each other's duals using a counting argument. We count the number of edges needed to connect all red and blue regions, and use Euler's formula to show the number of red faces must be equal to the number of blue vertices, and vice versa. The other direction follows from Lemma 1. Having two dual spanning trees (e.g., Fig. 3), simply draw the two spanning trees and for any cell not yet having a blue (red) piece add a crossing-free connection to the blue (red) polygon. \square

3.2 Spiderweb Gadgets

Let us now extend the result in the previous section, by showing how to include purple regions with more than four adjacent regions. For every purple region P with $\kappa(P) > 4$ we construct a *spiderweb* gadget and insert it into the graphs G_r and G_b , such that an argument similar to Lemma 2 can be applied.

A spiderweb gadget W of P with $\kappa(P)/2 = k$ red and k blue alternating adjacent regions consists of $\lfloor k/2 \rfloor + 1$ levels, labeled 0 (outermost) to $\lfloor k/2 \rfloor$ (innermost), see Fig. 4. Each level, except 0 and $\lfloor k/2 \rfloor$, is a cycle of k vertices. The level 0 has k (blue) vertices without any edges between them, and the innermost level $\lfloor k/2 \rfloor$ consists of only a single vertex. The vertices of even levels are blue and labeled with even numbers from 0 to $2k - 2$ clockwise. The vertices of odd levels are red and labeled with odd numbers 1 to $2k - 1$ clockwise.

Each vertex of level ℓ with $2 \leq \ell < \lfloor k/2 \rfloor$ is connected to the vertex with the same label on level $\ell - 2$. The single vertex of level $\lfloor k/2 \rfloor$ is connected to all the vertices of level $\lfloor k/2 \rfloor - 2$. This gives us $2k$ paths starting from levels 0 and 1 to the two innermost levels. We call these paths *spokes*, and refer to them by the label of the corresponding vertices. We embed the two resulting connected components in such a way that they are each other's dual, by making sure that we get a proper clockwise numbering on the vertices of the two outermost levels (see Fig. 4). The vertices on levels 0 and 1 represent respectively the blue and red regions around the purple region P and respect the adjacency order around P .

If a blue (or red) region touches P multiple times, then the corresponding vertices on level 0 (or 1) map to the same region and are in fact one and the same

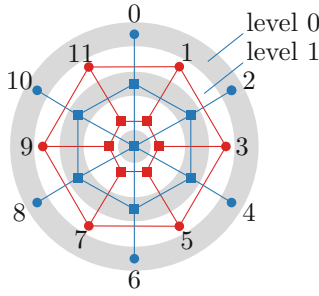


Fig. 4. Spiderweb gadget for $k = 6$: three blue levels with indices 0, 2, 4, and two red levels with indices 1, 3. (Color figure online)

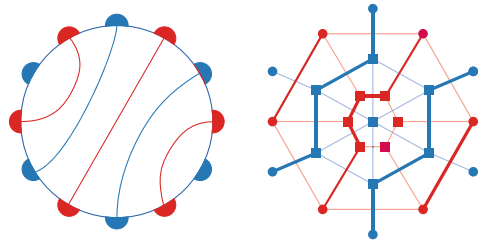


Fig. 5. Topology of the connections in a purple region and the corresponding bridging paths through a spiderweb gadget. (Color figure online)

vertex in G_b (or G_r). All edges connected to this vertex are consistent with the topology of the nested neighboring regions of P ; they intersect the same edges as they would when they were represented by multiple vertices.

To prove that all possible connections in P , which can occur in a painting Π , can be replicated in a spiderweb gadget W , we define *bridging paths*: let u and v be two vertices on level 0 in W that represent two blue regions that are connected by a painting Π through P . Assume that the clockwise distance from u to v is not greater than k , that is, if u has label x then v has label $(x + 2i) \bmod 2k$ for some $1 \leq i \leq \lfloor k/2 \rfloor$. To connect u and v with a bridging path, we start from u , go to level $2\lfloor(i + 1)/2\rfloor$ along the spoke x , take a shortest path within the level $2\lfloor(i + 1)/2\rfloor$ from the vertex with label x to the vertex with label $(x + 2i) \bmod 2k$, and move along the spoke $(x + 2i) \bmod 2k$ to vertex v . If there are two possible shortest paths, we take the clockwise path.

The same kind of path can be constructed for a pair of red vertices, but starting from level 1, going to level $2\lfloor i/2 \rfloor + 1$, and moving back to level 1. We now show that connecting different blue and red regions using bridging paths within the spiderweb gadgets results in blue trees and red trees, such that no pair of a blue and a red edge intersect (see Fig. 5 for an example).

By performing a case analysis on the possible red and blue pairs of adjacent regions to be connected, we can prove that the following lemma holds.

Lemma 3. *Consider a painting Π in which two blue and two red regions, adjacent to a purple region P , are connected through P . The corresponding vertices in the spiderweb gadget W of P can be connected by non-intersecting bridging paths.*

With spiderweb gadgets and the above lemma, we now strengthen Lemma 2 to the following lemma, without a condition on κ , and prove it in a similar way.

Lemma 4. *A 2-colored grid Γ in which each purple region has no holes admits a painting if and only if the corresponding G_r and G_b are each other’s exact duals.*

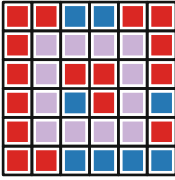


Fig. 6. An annulus-type purple region with adjacent blue and red regions, both inside and outside. (Color figure online)

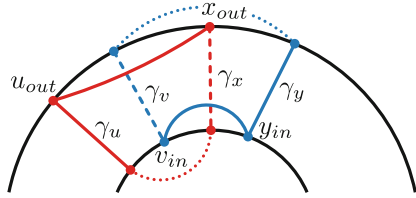


Fig. 7. By adding edges (v_{in}, y_{in}) and (u_{out}, x_{out}) we reconnect the disconnected subpolygons formed by removing cross-annulus connections γ_v and γ_x . (Color figure online)

3.3 Purple Regions with Holes

We may also have purple regions with holes (see Fig. 6). We show that the number of holes can be reduced without affecting the solvability. For simplicity of explanation we assume a region with a single hole (an annulus); regions with more holes can be reduced by considering only connections to the outer boundary.

Let P be a purple annulus. Any painting subdivides P into a number of colored simple components. Each component of color c connects one or more regions of color c on the boundary of P . The existence of a painting is defined only by the connectivity structure of these components. The connectivity of a component can be represented (transitively) using a set of non-intersecting simple paths (*connections*) each connecting two regions on the boundary. Let a *cross-annulus connection* γ_x be a connection between a region x_{in} on the inside of the annulus and a region x_{out} on the outside of the annulus. A (connectivity) *structure* is a maximal set of (pairwise non-intersecting) connections in P that can be extended to a valid painting. Let C_S be the set of cross-annulus connections in a given structure S . We assume the annulus is not degenerate, so red and blue regions exist both inside and outside the annulus.

Lemma 5. *If a structure S exists with two adjacent cross-annulus connections γ_x and γ_y of the same color, possibly separated by non-crossing connections, then there also exists a structure S' where $C_{S'} = C_S \setminus \{\gamma_y\}$.*

Lemma 6. *If there exists a structure S with $|C_S| > 3$ and all cross-annulus connections are alternating in color, then there also exists a structure S' with $|C_{S'}| - 2$ cross-annulus connections.*

Proof. Let $\gamma_u, \gamma_v, \gamma_x,$ and γ_y be four consecutive cross-annulus connections. W.l.o.g., assume γ_u and γ_x are red and γ_v and γ_y are blue. We remove γ_v and γ_x from the structure separating both the red and blue into two components. For both colors one component is still connected to the remaining cross-annulus connection γ_u , respectively γ_y . The disconnected components cannot both be

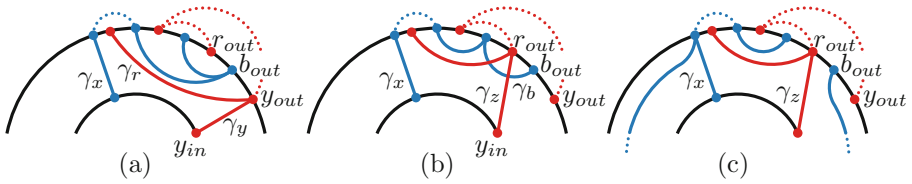


Fig. 8. (a) Initial configuration with several connections covering r_{out} . (b) Rerouting the blue connections, introducing γ_z , and rerouting the intersecting red connection leaves only one intersecting (blue) connection. (c) As the blue disconnected component cannot be covered by the new red connection, we can always connect it back to γ_x . (Color figure online)

on the outside (inside) of the annulus. If so, the connection γ_u to γ_x must be connected through x_{in} , and γ_v to γ_y through v_{in} . However, as there is no cross-annulus connection between γ_u and γ_y , any connection from γ_u to x_{in} separates γ_y and v_{in} . Hence, both connections cannot exist at the same time. The red and blue disconnected components are thus on different sides of the annulus and we connect them to γ_u respectively γ_y without mutually interfering (see Fig. 7). \square

Corollary 1. *If a structure exists, then a structure also exists that has exactly one red and one blue connection across each annulus.*

Lemma 7. *If a structure exists, then there also exists a structure with exactly one red and one blue cross-annulus connection starting from any two regions on the inner annulus and connecting to any two regions on the outer annulus.*

Proof. Let an *interval* be a maximal arc of the same color on the boundary. By Corollary 1 we know there exists a structure with exactly one red and one blue connection across the annulus. Let γ_x be the blue cross-annulus connection and γ_y the red cross-annulus connection. We show that each of the endpoints of the cross-annulus connection can freely be moved. W.l.o.g., assume that γ_x is not counter-clockwise adjacent to γ_y on the outside of the annulus. Let k_{out} , l_{out} , and m_{out} be three intervals in clockwise order on the outer boundary of the annulus. We say a clockwise connection through the annulus from k_{out} to m_{out} covers l_{out} .

Let b_{out} be the blue interval that is counter-clockwise adjacent to y_{out} and r_{out} the red interval that is counter-clockwise adjacent to b_{out} . Interval b_{out} may have several incoming blue connections that cover r_{out} (see Fig. 8(a)). We can rewire the blue connections inside the annulus to connect the blue intervals in sorted order around the annulus, resulting in only one blue connection γ_b that covers r_{out} . Similarly we can also rewire the red connections covering r_{out} , and ending at y_{out} , to ensure only one red connection γ_r covers r_{out} .

Remove γ_y and insert a new red cross-annulus connection $\gamma_z = (y_{in}, r_{out})$. The connection γ_z can only intersect γ_r and γ_b . Removing γ_r results in two red components, one of which contains γ_z . Assume w.l.o.g. that y_{out} is in the same

connected component as γ_z . As γ_r intersected γ_z , the disconnected component can be connected to γ_z while only intersecting γ_b (see Fig. 8(b)).

Removing γ_b results in two blue components, one of which contains γ_x . We prove that b_{out} must be part of the blue component not containing γ_x . Assume to the contrary that b_{out} is still connected to γ_x . Interval r_{out} must be connected to y_{out} outside of the annulus as there was only one red cross-annulus connection and γ_b blocked any connection through the inside of the annulus. Similarly, interval b_{out} must have been connected through the outside of the annulus, as it is separated from any other region inside the annulus by γ_y and γ_z . However, they cannot both be connected through the outside of the annulus, as the connection r_{out} to y_{out} separates b_{out} and x_{out} on the outside of the annulus. Contradiction.

Therefore, we can safely reconnect the disconnected blue component through the annulus to γ_x (see Fig. 8(c)). Repeatedly moving the end-point of one of the cross-annulus connections allows the creation of any configuration of the two red and blue cross-annulus connections without invalidating the structure. \square

Lemma 7 implies that we can cut the annulus open to reduce the number of holes of a purple region by one without changing the solvability of the problem. Together with Lemma 4, this then implies Theorem 1.

4 Optimizing Panels

As shown, not all colored grids admit a painting. Here we investigate the design of the panels themselves, assuming that some painting is possible. To this end, we define the complexity of a panel as the number of *pieces* of maximal red and blue areas in the panel, see Fig. 9. The complexity of a painting is the maximal complexity of any of its panels. A t -panel (t -painting) has complexity t .

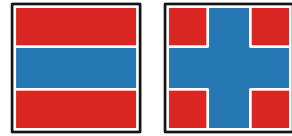


Fig. 9. Panels with complexity 3 and 5 respectively. (Color figure online)

Assuming some painting exists, we prove in this section that a 5-painting exists in general and that even a 2-painting exists if there are no white cells.

4.1 Ensuring a 5-Painting

We prove here that a 5-painting can always be realized. To this end, we show that a valid painting for a colored grid can be redrawn to include no more than three colored intervals along each side of all panels.

Lemma 8. *If a 2-colored grid admits a painting, then it admits a painting where each panel π has at most 3 intervals of alternating red and blue along each side.*

Proof (sketch). Assume that a panel π has at least 4 intervals of alternating red and blue on the left-side of π . As the painting is valid, both blue (/red) intervals are connected in the painting. For each interval we identify whether

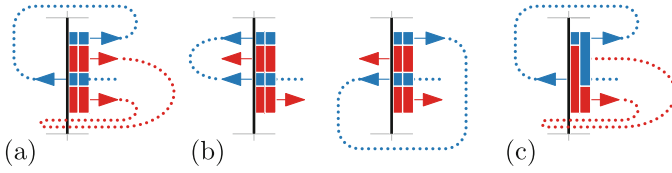


Fig. 10. Reducing the number of intervals along a side of panel π , where there are at least four. (b) The two middle directions cannot be the same, as we cannot connect them with nonintersecting paths. (c) Shortcutting inside π reduces the number of intervals while maintaining a painting. (Color figure online)

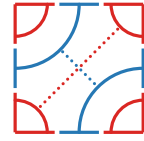


Fig. 11. A panel with six pieces can always be reduced to have five, using either dotted line. (Color figure online)

the path exiting or entering π connects to the other interval of the same color (see Fig. 10(a)). The red and blue path cannot leave or exit the border of π in the same direction for the middle two intervals (see Fig. 10(b)). To reduce the number of intervals, we recolor the interval by shortcutting both the blue and the red piece inside π (see Fig. 10(c)). \square

Theorem 2. *If a partially 2-colored grid admits a painting, then it admits a 5-painting.*

Proof. By Lemma 8 there are at most three alternately colored intervals along each side of π . If a red and blue interval meet in a corner, we extend one in π around the corner to get four intervals and use Lemma 8 to reduce it back to at most three. If we have more than five pieces, a piece that has only one interval in π can be removed while maintaining a painting. Each remaining piece connects at least two intervals: with k intervals, the number of pieces is at most $\lfloor k/2 \rfloor$. A 6-panel thus requires 12 intervals: four equal-color (red) corners and a middle interval (blue) along each side. This enforces two pieces between the blue intervals, and one in each corner. However, we can now reduce the number of pieces to five, connecting either two blue pieces or two red corners (Fig. 11). \square

This bound is tight as a 5-panel may be required when the grid includes white cells (Fig. 12(a)). A 5-panel with at least two pieces of each color is never required—though such a 4-panel may be necessary (Fig. 12(b)). The above proof implies that there is only one option to create such a 5-panel: it has only two ways to connect the two blue pieces; both can be simplified to a 4-panel (Fig. 13).

4.2 Ensuring a 2-Painting

We show that a fully 2-colored grid (rectangular and without white cells) even admits a 2-painting, provided it admits any painting. As an intermediate step, we first prove that a painting exists that uses only one blue piece in any panel.

Lemma 9. *If a fully 2-colored grid admits a painting, then it admits a painting in which each panel has at most one blue piece.*

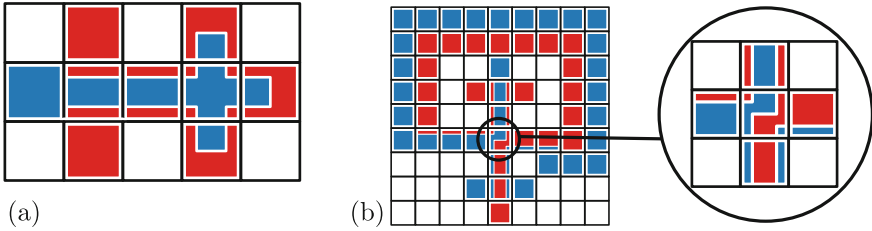


Fig. 12. Examples requiring complex panels. (a) A colored grid requiring a 5-panel. (b) A colored grid requiring a 4-panel with two pieces of both colors in the same cell. (Color figure online)

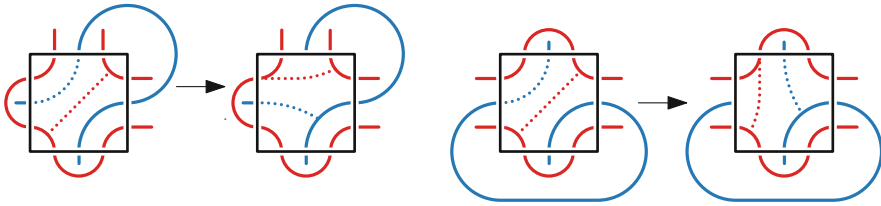


Fig. 13. There are two configurations for a 5-panel where both colors have at least two pieces. Both possible configuration can be simplified to a 4-panel. (Color figure online)

Proof (sketch). Since the grid admits a painting, we show how to modify the painting of each purple region P to ensure that the lemma holds. We first create a blue spanning forest in the panels of P that connects the panel-centers of adjacent panels. This ensures that each panel has exactly one blue piece inside, but may result in a disconnected blue polygon. However, since we know that a painting exists and the current solution maps to some forest in G_b , Lemma 1 implies that its dual G_r has a cycle around some tree in the forest. Hence, we can add connections between unconnected subpolygons to create a single blue polygon again, without disconnecting the red polygon. \square

The above construction relies on the alternation of the blue and red intervals along the boundary of P . As there are no white cells we can guarantee this alternating pattern. Indeed, the higher complexity with white cells is caused by long connections along a purple region’s boundary that are needed to achieve this alternating pattern for a partially colored grid (e.g., Fig. 12).

Theorem 3. *If a fully 2-colored grid admits a painting, then it admits a 2-painting.*

Proof (sketch). Since the fully 2-colored grid admits a painting, Lemma 9 implies that there is a painting Π where the panel for every purple cell contains only a single blue piece. For any purple cell with more than one red piece, we remove red pieces that only connect to one neighboring panel and recolor red corners

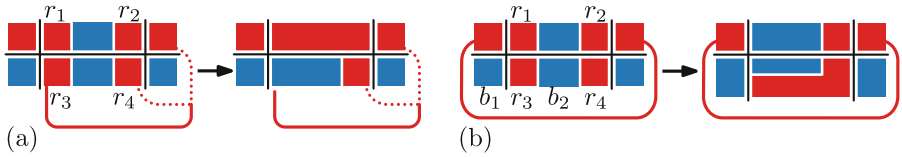


Fig. 14. Reducing panel complexity when there are two red corners along the same panel side. (a) The corners are connected via adjacent (or the same) sides of the panel: connect r_1 and r_2 , and recolor r_3 to blue. (b) The corners are connected via opposite sides: recolor r_1 to blue and connect r_3 and r_4 as well as b_1 and b_2 . (Color figure online)

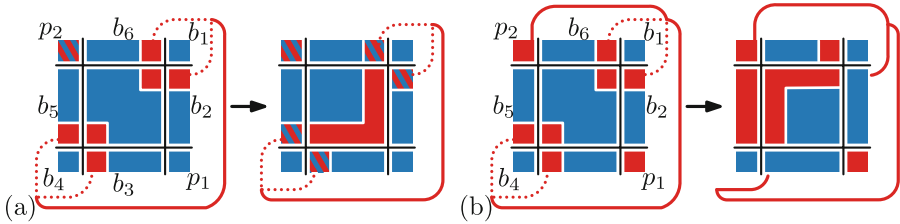


Fig. 15. Two diagonally positioned red corners. The complexity of the panel can be reduced by introducing a red L -shape that connects all the red. (a) Reducing complexity if either p_1 or p_2 was blue. (b) Reducing complexity if both p_1 and p_2 were red. (Color figure online)

to blue if the other three cells incident to that corner have a red corner as well. Now, the pattern of the panel matches one of the following four cases.

1. There are two red corners r_1 and r_2 on the same side of the panel. The connecting path exits the current panel via the same side and enters either on the same or adjacent side (see Fig. 14(a)).
2. There are two red corners r_1 and r_2 on the same side of the panel. The connecting path exits the panel via opposite sides of the panel (see Fig. 14(b)). The blue piece connects only downwards in the panel below.
3. There are two red corners r_1 and r_2 that do not share a common side of the panel. In this case the other corners are blue, otherwise one of the two previous cases applies (see Fig. 15(a)). Furthermore, either p_1 or p_2 is blue.
4. There are two red corners r_1 and r_2 that do not share a common side of the panel (see Fig. 15(b)). Furthermore, both p_1 and p_2 are red.

We design a reduction rule for each case, as sketched in Figs. 14 and 15. Repeated application of the reduction rules, interlaced with the reduction of the number of red pieces in a panel, results in a 2-painting. \square

5 Conclusion

We took the first steps towards investigating a disjoint-polygons representation for visualizing set memberships (hypergraphs). We investigated the 2-color

version in which each element is positioned as a cell in a (unit-)grid. We showed how to test whether a disjoint-polygons representation is possible for a given 2-colored grid. Moreover, we proved that if such a representation is possible, then we can also bound the complexity of the corresponding “panels” (the coloring of a single cell). Each panel requires at most five colored pieces, and even only two pieces are sufficient when no white cells are present in the grid.

There are myriad options for further exploration. As not all grids admit a painting, we could study minimizing the number of polygons of the same color. We have not touched upon variants with more colors: does our approach readily generalize? However, considering the restrictions already in the studied 2-color variant, it seems likely that many practical instances do not admit a painting. If we allow rearranging elements, the 2-color variant becomes trivial, but is particularly interesting for multiple colors. Finally, we may consider the situation where some cells have no assigned set of colors but may be painted using any subset of the colors. Given enough such cells, the disjoint-polygons encoding can then represent more than Zykov-planar hypergraphs but cannot represent all planar supports.

Acknowledgments. The authors would like to thank Jason Dykes for fruitful discussions at an early stage of this research.

References

1. Akitaya, H.A., Löffler, M., Tóth, C.D.: Multi-colored spanning graphs. In: Hu, Y., Nöllenburg, M. (eds.) GD 2016. LNCS, vol. 9801, pp. 81–93. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_7
2. Alper, B., Riche, N.H., Ramos, G., Czerwinski, M., Czerwinski, M.: Design study of LineSets, a novel set visualization technique. *IEEE Trans. Vis. Comput. Graph.* **17**(12), 2259–2267 (2011)
3. Alsallakh, B., Aigner, W., Miksch, S., Hauser, H.: Radial sets: interactive visual analysis of large overlapping sets. *IEEE Trans. Vis. Comput. Graph.* **19**(12), 2496–2505 (2013)
4. Alsallakh, B., Micallef, L., Aigner, W., Hauser, H., Miksch, S., Rodgers, P.: The state of the art of set visualization. *Comput. Graph. Forum* **35**(1), 234–260 (2016)
5. Baron, M.: A note on the historical development of logic diagrams: Leibniz, Euler and Venn. *Math. Gaz.* **53**(384), 113–125 (1969)
6. Buchin, K., van Kreveld, M., Meijer, H., Speckmann, B., Verbeek, K.: On planar supports for hypergraphs. *J. Graph Algorithms Appl.* **15**(4), 533–549 (2011)
7. Collins, C., Penn, G., Carpendale, S.: Bubble sets: revealing set relations with isocontours over existing visualizations. *IEEE Trans. Vis. Comput. Graph.* **15**(6), 1009–1016 (2009)
8. Dinkla, K., van Kreveld, M., Speckmann, B., Westenberg, M.: Kelp diagrams: point set membership visualization. *Comput. Graph. Forum* **31**(3pt1), 875–884 (2012)
9. Eppstein, D., Italiano, G., Tamassia, R., Tarjan, R., Westbrook, J., Yung, M.: Maintenance of a minimum spanning forest in a dynamic plane graph. *J. Algorithms* **13**(1), 33–54 (1992)
10. Riche, N.H., Dwyer, T.: Untangling Euler diagrams. *IEEE Trans. Vis. Comput. Graph.* **16**(6), 1090–1099 (2010)

11. Hurtado, F., Korman, M., van Kreveld, M., Löffler, M., Sacristán, V., Silveira, R.I., Speckmann, B.: Colored spanning graphs for set visualization. In: Wismath, S., Wolff, A. (eds.) GD 2013. LNCS, vol. 8242, pp. 280–291. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03841-4_25
12. Kaufmann, M., van Kreveld, M., Speckmann, B.: Subdivision drawings of hypergraphs. In: Tollis, I.G., Patrignani, M. (eds.) GD 2008. LNCS, vol. 5417, pp. 396–407. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00219-9_39
13. Mäkinen, E.: How to draw a hypergraph. *Int. J. Comput. Math.* **34**, 177–185 (1990)
14. Meulemans, W., Dykes, J., Slingsby, A., Turkay, C., Wood, J.: Small multiples with gaps. *IEEE Trans. Vis. Comput. Graph.* **23**(1), 381–390 (2017)
15. Meulemans, W., Riche, N.H., Speckmann, B., Alper, B., Dwyer, T., Dwyer, T.: KelpFusion: a hybrid set visualization technique. *IEEE Trans. Vis. Comput. Graph.* **19**(11), 1846–1858 (2013)
16. Purchase, H.: Metrics for graph drawing aesthetics. *J. Vis. Lang. Comput.* **13**(5), 501–516 (2002)
17. Purchase, H.C., Cohen, R.F., James, M.: Validating graph drawing aesthetics. In: Brandenburg, F.J. (ed.) GD 1995. LNCS, vol. 1027, pp. 435–446. Springer, Heidelberg (1996). <https://doi.org/10.1007/BFb0021827>
18. Simonetto, P., Auber, D.: Visualise undrawable Euler diagrams. In: Proceedings of the 12th Conference on Information Visualisation, pp. 594–599 (2008)
19. Simonetto, P., Auber, D., Archambault, D.: Fully automatic visualisation of overlapping sets. *Comput. Graph. Forum* **28**(3), 967–974 (2009)
20. Tutte, W.: *Graph Theory*. Addison-Wesley, Menlo Park (1984)
21. van Goethem, A., Kostitsyna, I., van Kreveld, M., Meulemans, W., Sondag, M., Wulms, J.: The painter's problem: covering a grid with colored connected polygons. Computing Research Repository, [arXiv:1709.00001](https://arxiv.org/abs/1709.00001) (2017)
22. Walsh, T.: Hypermaps versus bipartite maps. *J. Comb. Theor.* **18**, 155–163 (1975)
23. Wood, J., Dykes, J.: Spatially ordered treemaps. *IEEE Trans. Vis. Comput. Graph.* **14**(6), 1348–1355 (2008)

Triangle-Free Penny Graphs: Degeneracy, Choosability, and Edge Count

David Eppstein^(✉)

Department of Computer Science, University of California, Irvine, USA
eppstein@uci.edu

Abstract. We show that triangle-free penny graphs have degeneracy at most two, list coloring number (choosability) at most three, diameter $D = \Omega(\sqrt{n})$, and at most $\min(2n - \Omega(\sqrt{n}), 2n - D - 2)$ edges.

1 Introduction

Penny graphs are the contact graphs of unit circles [1, 2] — they are formed from non-overlapping sets of unit circles by creating a vertex for each circle and an edge for each tangency between two circles — and as such, fit into a long line of graph drawing research on contact graphs of geometric objects [3–7]. The same graphs (except the graph with no edges) are also proximity graphs, the graphs determined from a finite set of points in the plane by adding edges between all closest pairs of points, and for this reason they are also called minimum-distance graphs [8, 9]. A minimum-distance representation can be obtained from a contact representation by choosing a point at the center of each circle, and a contact representation can be obtained from a minimum-distance representation by scaling the points so their minimum distance is two and using each point as the center of a unit circle. However, finding either type of representation given only the graph is NP-hard, even for trees [10].

As graph drawings, minimum distance representations are in many ways ideal: they have no crossings, all edges have unit length, and the angular resolution is at least $\pi/3$. Every graph that can be drawn with this combination of properties is a penny graph. Moreover, penny graphs have *degeneracy* at most three, where the degeneracy of a graph G is the minimum number d such that every subgraph of G contains a vertex of at most d . Equivalently, the vertices of any penny graph can be ordered so each vertex has at most three neighbors later than it in the ordering. This ordering leads to a linear-time greedy 4-coloring algorithm [11], much simpler than known quadratic-time 4-coloring algorithms for arbitrary planar graphs [12]. Additionally, although planar graphs with n vertices can have $3n - 6$ edges, penny graphs have at most $\lfloor 3n - \sqrt{12n - 3} \rfloor$ edges [13]. This bound is tight for pennies tightly packed into a hexagon [14], and

Supported in part by the National Science Foundation under Grants CCF-1228639, CCF-1618301, and CCF-1616248.

its lower-order square-root term stands in an intriguing contrast to many similar bounds on the edge numbers of planar graphs, k -planar graphs, quasi-planar graphs, and minor-closed graph families, with constant or unknown lower-order terms [15–20].

Swanepoel [9] first considered corresponding problems for the *triangle-free* penny graphs. In graph drawing terms, these are the graphs that can be drawn with no crossings, unit-length edges, and angular resolution strictly larger than $\pi/3$. Swanepoel observed that, as with triangle-free planar graphs more generally, an n -vertex triangle-free penny graph can have at most $2n - 4$ edges. As a lower bound, the square grids have $\lfloor 2n - 2\sqrt{n} \rfloor$ edges, as do some subsets of grids and some pentagonally-symmetric graphs found by Oloff de Wet [9]. Swanepoel conjectured that, of the two bounds, it is the lower bound that is tight.

Triangle-free planar graphs more generally have also been considered. Grötzsch proved that these graphs are 3-colorable [21,22] and they can be 3-colored in linear time [23]. However, not every triangle-free planar graph is 3-list-colorable: if each vertex is given a list of three colors, it is not always possible to assign each vertex a color from its list that differs from all its neighbors' assigned colors [24]. 3-list-colorability is known for bipartite planar graphs [25], planar graphs with girth at least five [22], and planar graphs of girth four with well-separated 4-cycles [26], but these subclasses do not include all triangle-free penny graphs.

We continue these lines of research with the following new results.

- Every triangle-free penny graph with at least one cycle has at least four vertices of degree two or less. Consequently, the triangle-free penny graphs have degeneracy at most two.
- Every triangle-free penny graph has list chromatic number (choosability) at most three, and any list-coloring problem on a triangle-free penny graph with three colors per vertex can be solved in linear time.
- Every n -vertex triangle-free penny graph has at most $2n - \Omega(\sqrt{n})$ edges. Thus, the form of Swanepoel's conjectured edge bound is correct, although we cannot confirm the conjectured constant factor on the square-root term.
- Every penny graph has graph-theoretic diameter $\Omega(\sqrt{n})$, and every triangle-free penny graph with n vertices and diameter D has at most $2n - D - 2$ edges. The combination of these two results provides an alternative proof of the $2n - \Omega(\sqrt{n})$ edge bound, but with a worse constant factor in the Ω .

2 Degeneracy

We begin by showing that every triangle-free penny graph with at least one cycle has at least four vertices of degree two or less. It is convenient to begin with a special case of these graphs, the ones that are biconnected.

Lemma 1. *Every biconnected triangle-free penny graph has at least four vertices of degree two.*

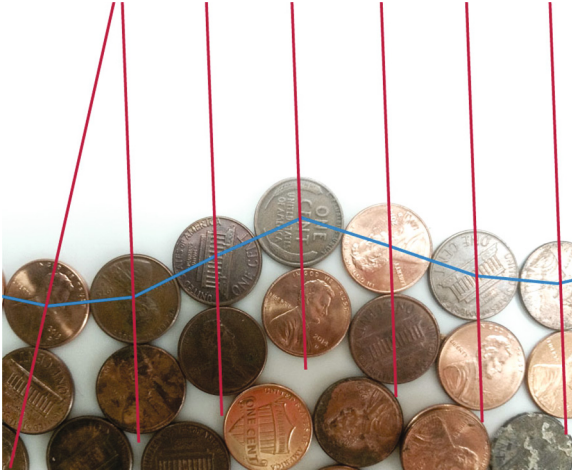


Fig. 1. Rays R_v extending from the center of each boundary vertex directly away from the clockwise neighbor of its clockwise boundary neighbor, used in the proof of Lemma 1.

Proof. Given a biconnected triangle-free penny graph G , and its representation as a penny graph, the outer face of the representation (as in any biconnected plane graph) consists of a simple cycle of vertices; in particular each vertex of this face has at least two neighbors. For each vertex v of this simple cycle, let w be the clockwise neighbor of v in the cycle, and let u be the neighbor of v that is next in clockwise order around v from w ; define a ray R_v , having the center of the disk of v as its apex, and pointing directly away from the center of u (Fig. 1). Given the same boundary vertices v and w in clockwise order, define the angle θ_w to be the angle made by rays R_v and R_w , assigned a sign so that θ_w is positive if R_w turns a clockwise angle (less than $\pi/2$) from R_v , and negative if R_w turns counterclockwise with respect to R_v . If R_v and R_w are parallel, then we define $\theta_w = 0$. Then these rays and their angles have the following properties:

- Each ray R_v points into the outer face of the drawing. Therefore, the sum of the turning angles of the rays as we traverse the entire outer face in clockwise order, $\sum \theta_v$, must equal 2π .
- If a boundary vertex w has degree three or more, then $\theta_w \leq 0$. For, if v and w are consecutive on the outer face, with R_v pointing away from a neighbor u of v (as above) and R_w pointing away from a neighbor x of w , then the assumption that w has degree at least three implies that $x \neq v$, and the assumption that G is triangle-free implies that $x \neq u$. If x and u touch, so that $uvw x$ forms a quadrilateral in G , then R_v and R_w are necessarily parallel, so $\theta_w = 0$. In any other case, to prevent x and u from touching, x must be rotated counterclockwise around w from the position where it would touch u , causing angle θ_w to become negative.

- At a boundary vertex w of degree two, $\theta_w < 2\pi/3$. For, in this case, R_w points away from v , the counterclockwise neighbor of w on the outer face. Let u be the neighbor of v such that R_v points away from u ; then $w \neq u$. Because both R_v and R_w belong to lines through the center of v , their angle θ_w is complementary to angle wvu , which must be greater than $\pi/3$ in order to prevent circles u and w from overlapping or touching (and forming a triangle). Therefore, θ_w is less than $2\pi/3$.

For the sequence of angles θ_w , each less than $2\pi/3$, to add to a total angle of 2π , there must be at least four positive angles in the sequence, and therefore there must be at least four degree-two vertices. \square

Theorem 1. *Every triangle-free penny graph G with at least one cycle has at least four non-articulation vertices of degree two or less.*

Proof. By the assumption that G has at least one cycle, it has at least one nontrivial biconnected component C . By Lemma 1, C has at least four degree-two vertices, each of which either has degree two in G or forms an articulation point of G . If it forms an articulation point, then the tree of biconnected components connected through it to G has at least one leaf, which must either be a vertex of degree one in G or a nontrivial biconnected component with at least four degree-two vertices, only one of which can be an articulation point. Thus, each of the four degree-two vertices in C is either itself a non-articulation vertex of degree at most two in G or leads to such a vertex. \square

The bound on the number of degree-two vertices is tight for square grids.

Theorem 2. *The degeneracy of every triangle-free penny graph is at most two.*

Proof. Every subgraph of a triangle-free penny graph is another triangle-free penny graph, so the result follows from Theorem 1 and from the fact that, in a graph with no cycles (a forest) there always exists a vertex of degree one or less (a leaf or an isolated vertex). \square

3 Choosability

The *choosability*, or *list chromatic number*, of a graph G is the minimum number c such that, for every labeling of each vertex of G by a list of c colors, there exists an assignment of a single color from its list to each vertex, with no two adjacent vertices assigned the same color. The usual graph coloring problem is a special case in which all vertices have the same list. Known relations between list coloring and graph degeneracy [25] give us the following result:

Theorem 3. *If a triangle-free penny graph is labeled by a list of three colors for each vertex, then we can find a solution to the list coloring problem for the resulting labeled graph in linear time. The algorithm needs as input only the abstract graph, not its representation as a penny graph.*

Proof. Find a vertex of degree at most two, remove it from the graph, color the remaining subgraph recursively, and put back the removed vertex. It has at most two neighbors, preventing it from being assigned at most two colors from its list of three colors, so there always remains at least one color available for it to use.

Linear time follows by maintaining the degree of each vertex in the reduced graph formed by the removals, a list of vertices of reduced degree at most two, and a stack of removals to be reversed. It takes constant time per vertex removal and replacement to update these data structures. \square

Corollary 1. *Triangle-free penny graphs have choosability at most three.*

This bound is tight as the odd cycles of length ≥ 5 are triangle-free penny graphs with choosability exactly three.

4 Edge Count

We derive a bound on the number of edges of a triangle-free penny graph by using the isoperimetric theorem to show that the outer face of any representation as a penny graph has many vertices, and then by using Euler's formula to show that a planar graph with a large face has few edges.

Lemma 2. *Let v be a vertex of a penny graph that (in some representation of the graph as a penny graph) is not on the outer face. Then, in the Voronoi diagram of the centers of the circles in the representation, the Voronoi cell containing v has area at least $2\sqrt{3}$, which is the area of a regular hexagon circumscribed around a unit circle.*

Proof. The area is minimized when each Voronoi neighbor of v is as close as possible to v (so that the neighbor's circle touches that of v , causing the Voronoi cell of v to circumscribe its circle), when the neighbors are equally spaced around v (forming a regular polygon), and when the number of neighbors is as large as possible (forming a hexagon). The first two of these claims follow from the fact that any other configuration of neighbors can be continuously deformed to make the area of v 's cell smaller, while the last one follows by comparing the areas of the other possible regular polygons. \square

Lemma 3. *In any penny graph representation of a graph G with n vertices, the number of vertex-face incidences on the outer face of the representation is at least*

$$\sqrt{\pi \cdot 2\sqrt{3} \cdot n} - O(1) \approx 3.3\sqrt{n}.$$

Proof. Unless there are at least this many incidences, by Lemma 2 there must be a total area of at least $2\sqrt{3} \cdot n - O(\sqrt{n})$ enclosed by the outer face, because each Voronoi cell of an inner vertex is enclosed and the Voronoi cells are all disjoint. The result follows from the facts that each vertex-face incidence accounts for 2 units of length of the outer face (the two radii of a single unit circle in the representation, along which the outer face enters and then leaves that circle) and that any curve that encloses area A must have length at least $2\sqrt{\pi A}$ (the isoperimetric theorem, with the shortest enclosing curve being a circle). \square

Lemma 4. *Let G be an n -vertex triangle-free plane graph in which one face has k vertex-face incidences. Then G has at most $2n - k/2 - 2$ edges.*

Proof. Vertex-face incidences and edge-face incidences on any face are equal, so the same face of G that has k vertex-face incidences also has k edge-face incidences. We count the number of edge-face incidences in G in two ways: by counting two incidences for each edge, and by summing the lengths of the faces. Each face of G has at least four edges, so if there are e edges and f faces then we have the inequality $2e \geq 4(f - 1) + k$, or equivalently $e/2 - k/4 + 1 \geq f$. Using this inequality to replace f in Euler’s formula $n - e + f = 2$, we obtain $n - e + e/2 - k/4 + 1 \geq 2$, or equivalently $e \leq 2n - k/2 - 2$ as claimed. \square

Theorem 4. *The number of edges in any n -vertex triangle-free penny graph is at most*

$$2n - \frac{1}{2}\sqrt{\pi \cdot 2\sqrt{3} \cdot n} + O(1) \approx 2n - 1.65\sqrt{n}.$$

Proof. Lemma 3 proves the existence of a large face, and plugging the size of this face as the variable k in Lemma 4 gives the stated bound. \square

We leave the problem of closing the gap between this upper bound and Swanepoel’s $2n - 2\sqrt{n}$ lower bound as open for future research.

5 Diameter

Our results on degeneracy and number of edges can be connected via the following two results, which provide an alternative proof that the number of edges in a triangle-free penny graph is $2 - \Omega(\sqrt{n})$.

Theorem 5. *Every connected n -vertex penny graph has diameter $\Omega(\sqrt{n})$.*

Proof. By a standard isodiametric inequality [27], for the convex hull of n disjoint unit disks to enclose area $2\pi n$, it must have (geometric) diameter $\Omega(\sqrt{n})$. In order to connect two unit disks at geometric distance $\Omega(\sqrt{n})$ from each other, they must also be at graph-theoretic distance $\Omega(\sqrt{n})$. \square

Theorem 6. *Every connected n -vertex triangle-free penny graph G with diameter D has at most $2n - D - 2$ edges.*

Proof. We use induction on n . If G has no cycle, it is a tree, with $n - 1$ edges, and the result follows from the fact that $D \leq n - 1$. Otherwise, let uw be a diameter pair, and let v be any vertex of degree at most two, whose removal does not disconnect G , distinct from u and w . The existence of v follows from Theorem 1. Then $G - v$ has one less vertex, one or two fewer edges, and diameter at least D . The result follows by applying the induction hypothesis to $G - v$. \square

It is not true more generally that 2-degenerate triangle-free planar graphs with diameter D have at most $2n - D - 2$ edges; Theorem 6 relies on the specific properties of triangle-free penny graphs. However, we can prove analogous bounds of $2n - \Omega(\sqrt{n})$ and $2n - D - 2$ on the numbers of edges in *squaregraphs* [28], plane graphs in which every bounded face is a quadrilateral and every vertex that does not belong to the unbounded face has degree at least four. The details are given in the appendix of the preprint of this paper [29].

References

1. Hliněný, P., Kratochvíl, J.: Representing graphs by disks and balls (a survey of recognition-complexity results). *Discrete Math.* **229**(1–3), 101–124 (2001)
2. Pisanski, T., Randić, M.: Bridges between geometry and graph theory. In: Gorini, C.A. (ed.) *Geometry at Work. MAA Notes*, vol. 53, pp. 174–194. Cambridge University Press (2000)
3. de Fraysseix, H., Ossona de Mendez, P., Rosenstiehl, P.: On triangle contact graphs. *Comb. Probab. Comput.* **3**(2), 233–246 (1994)
4. Buchsbaum, A.L., Gansner, E.R., Procopiuc, C.M., Venkatasubramanian, S.: Rectangular layouts and contact graphs. *ACM Trans. Algorithms* **4**(1), A8 (2008)
5. Klawitter, J., Nöllenburg, M., Ueckerdt, T.: Combinatorial properties of triangle-free rectangle arrangements and the squarability problem. In: Di Giacomo, E., Lubiw, A. (eds.) *GD 2015. LNCS*, vol. 9411, pp. 231–244. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27261-0_20
6. Hliněný, P.: Contact graphs of line segments are NP-complete. *Discrete Math.* **235**(1–3), 95–106 (2001)
7. Alam, M.J., Eppstein, D., Kaufmann, M., Kobourov, S.G., Pupyrev, S., Schulz, A., Ueckerdt, T.: Contact graphs of circular arcs. In: Dehne, F., Sack, J.-R., Stege, U. (eds.) *WADS 2015. LNCS*, vol. 9214, pp. 1–13. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21840-3_1
8. Csizmadia, G.: On the independence number of minimum distance graphs. *Discrete Comput. Geom.* **20**(2), 179–187 (1998)
9. Swanepoel, K.J.: Triangle-free minimum distance graphs in the plane. *Geombinatorics* **19**(1), 28–30 (2009)
10. Bowen, C., Durocher, S., Löffler, M., Rounds, A., Schulz, A., Tóth, C.D.: Realization of Simply connected polygonal linkages and recognition of unit disk contact trees. In: Di Giacomo, E., Lubiw, A. (eds.) *GD 2015. LNCS*, vol. 9411, pp. 447–459. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27261-0_37
11. Hartshfield, N., Ringel, G.: Problem 8.4.8. In: *Pearls in Graph Theory: A Comprehensive Introduction*. Dover Books on Mathematics, pp. 177–178. Courier Corporation (2003)
12. Robertson, N., Sanders, D.P., Seymour, P., Thomas, R.: Efficiently four-coloring planar graphs. In: Miller, G.L. (ed.) *Proceedings of the 28th ACM Symposium on Theory of Computing (STOC 1996)*, pp. 571–575. Association for Computing Machinery (1996)
13. Harborth, H.: Lösung zu problem 664A. *Elemente der Mathematik* **29**, 14–15 (1974)
14. Kupitz, Y.S.: On the maximal number of appearances of the minimal distance among n points in the plane. In: Böröczky, K., Tóth, G.F. (eds.) *Intuitive Geometry: Papers from the Third International Conference. Colloq. Math. Soc. János Bolyai, Szeged, 2–7 September 1991*, vol. 63, pp. 217–244, North-Holland (1994)

15. Ackerman, E., Tardos, G.: On the maximum number of edges in quasi-planar graphs. *J. Comb. Theory Ser. A* **114**(3), 563–571 (2007)
16. Agarwal, P.K., Aronov, B., Pach, J., Pollack, R., Sharir, M.: Quasi-planar graphs have a linear number of edges. *Combinatorica* **17**(1), 1–9 (1997)
17. Brandenburg, F.J., Eppstein, D., Gleißner, A., Goodrich, M.T., Hanauer, K., Reislhuber, J.: On the density of maximal 1-Planar graphs. In: Didimo, W., Patrignani, M. (eds.) *GD 2012*. LNCS, vol. 7704, pp. 327–338. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36763-2_29
18. Eppstein, D.: Densities of minor-closed graph families. *Electron. J. Comb.* **17**(1), R136 (2010)
19. Pach, J., Tóth, G.: Graphs drawn with few crossings per edge. *Combinatorica* **17**(3), 427–439 (1997)
20. Suk, A., Walczak, B.: New bounds on the maximum number of edges in k -quasi-planar graphs. *Comput. Geom. Theory Appl.* **50**, 24–33 (2015)
21. Grötzsch, H.: Zur Theorie der diskreten Gebilde, VII: Ein Dreifarbensatz für dreikreisfreie Netze auf der Kugel. *Wiss. Z. Martin-Luther-U. Halle-Wittenberg Math. Nat. Reihe* **8**, 109–120 (1959)
22. Thomassen, C.: A short list color proof of Grötzsch’s theorem. *J. Comb. Theory Ser. B* **88**(1), 189–192 (2003)
23. Dvořák, Z., Kawarabayashi, K., Thomas, R.: Three-coloring triangle-free planar graphs in linear time. In: Mathieu, C. (ed.) *Proceedings of the 20th ACM-SIAM Symposium on Discrete Algorithms (SODA 2009)*, pp. 1176–1182. Society for Industrial and Applied Mathematics (2009)
24. Voigt, M.: A not 3-choosable planar graph without 3-cycles. *Discrete Math.* **146**(1–3), 325–328 (1995)
25. Alon, N., Tarsi, M.: Colorings and orientations of graphs. *Combinatorica* **12**(2), 125–134 (1992)
26. Dvořák, Z., Lidický, B., Škrekovski, R.: 3-choosability of triangle-free planar graphs with constraint on 4-cycles. *SIAM J. Discrete Math.* **24**(3), 934–945 (2010)
27. Bieberbach, L.: Über eine Extremaleigenschaft des Kreises. *Jber. Deutsch. Math. Verein.* **24**, 247–250 (1915)
28. Bandelt, H.J., Chepoi, V., Eppstein, D.: Combinatorics and geometry of finite and infinite squaregraphs. *SIAM J. Discrete Math.* **24**(4), 1399–1440 (2010)
29. Eppstein, D.: Triangle-Free penny graphs: degeneracy, choosability, and edge count. Electronic preprint [arXiv:1708.05152](https://arxiv.org/abs/1708.05152) (2017)

Beyond Planarity

1-Fan-Bundle-Planar Drawings of Graphs

Patrizio Angelini¹, Michael A. Bekos¹, Michael Kaufmann¹,
Philipp Kindermann²(✉), and Thomas Schneck¹

¹ Institut für Informatik, Universität Tübingen, Tübingen, Germany

² LG Theoretische Informatik, FernUniversität in Hagen, Hagen, Germany

philipp.kindermann@fernuni-hagen.de

Abstract. Edge bundling is an important concept heavily used for graph visualization purposes. To enable the comparison with other established near-planarity models in graph drawing, we formulate a new edge-bundling model which is inspired by the recently introduced fan-planar graphs. In particular, we restrict the bundling to the endsegments of the edges. Similarly to 1-planarity, we call our model *1-fan-bundle-planarity*, as we allow at most one crossing per bundle.

For the two variants where we allow either one or, more naturally, both endsegments of each edge to be part of bundles, we present edge density results and consider various recognition questions, not only for general graphs, but also for the outer and 2-layer variants. We conclude with a series of challenging questions.

1 Introduction

Edge bundling is a powerful tool used in information visualization to avoid visual clutter. When the edge density of the network is too high, the traditional techniques of graph layouts and flow maps become unusable. In this case, grouping together parts of edges that flow parallel to each other into a single bundle allows to reduce the clutter and improve readability. Among the many, we mention here the seminal papers of Holten [18], and of Telea and Ersoy [23], which focus on radial layouts, as well as works on flow maps [10] and parallel coordinates [25]. For an overview refer to Zhou et al. [24].

In this work, we combine for the first time this powerful visualization technique with previous theoretical considerations from the area of nearly-planar graphs, where in addition to a planar graph structure some crossings are allowed, if they are limited to locally defined configurations. Classical examples include *1-planar* graphs [22], which can be drawn so that each edge is crossed at most once, and *quasi-planar* graphs [2], which allow for drawings not containing any three mutually crossing edges.

Another typical example of nearly-planar graphs are the *fan-planar* graphs [20]. In a *fan-planar drawing* [6–8, 20], an edge is allowed to cross multiple edges as long as they belong to the same *fan*, that is, if they are all incident to a common vertex; refer to Fig. 1a. Such a crossing is called a *fan crossing*. The idea is that edges incident to the same vertex are somehow close to each other, and

thus having an edge crossing all of them does not affect readability too much. In other words, edges of a fan can be grouped into a *bundle*, so that the crossings between an edge and all the edges of the fan become a single crossing between this edge and the corresponding bundle. In Fig. 1b, we show the bundle-like edge routing corresponding to the fan-planar drawing in Fig. 1a. Note, however, that the original definition of fan-planar drawings does not always allow for this type of bundling, as in the case of graph $K_{4,n-4}$, for large enough n (see Sect. 2).

We thus introduce *1-fan-bundle-planar* (*1-fbp* for short) drawings, in which edges of a fan can be bundled together and crossings between bundles are allowed as long as each bundle is crossed by at most one other bundle; see Figs. 1b–d. Formally, in a 1-fbp drawing every edge has 3 parts: the first and last parts are *fan-bundles*, which may be shared by several edges; the middle part is *unbundled*. Each fan-bundle can cross at most one other fan-bundle. The unbundled parts are crossing-free. Fan-bundles are not allowed to branch, i.e., each fan-bundle has two endpoints: one of them is the vertex the fan is incident to, while at the other one all edges in the fan are separated from each other.

The “1-planarity” restriction prevents a fan-bundle of an edge to cross edges of several fans, which is not allowed in fan-planar drawings. Since every edge has two fan-bundles, each of which can cross another fan-bundle, it is possible that an edge crosses two different fans, making the drawing not fan-planar. To avoid this, we introduce a restricted model of 1-fbp drawings, called *1-sided*, in which an edge can be bundled with other edges only on one of its endvertices; see Fig. 1d. This restriction implies that 1-sided 1-fbp drawings are fan-planar. As we will see in Sect. 2, this is not the case for the so-called *2-sided* model, in which each edge has two fan-bundles (see Figs. 1b–c).

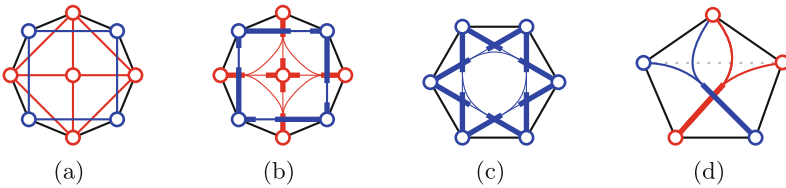


Fig. 1. (a–b) The fan-planar graph of (a) is redrawn in (b) under the 2-sided model, (c) a 2-sided 1-fbp drawing of K_6 , (d) a 1-sided 1-fbp drawing of $K_5 \setminus e$ (the missing edge is drawn dotted).

Since each bundle collects a set of edges and allows them to participate in a crossing, natural near-planarity theoretical questions arise: (i) Characterize or recognize the graphs that admit 1-fbp drawings, and (ii) provide upper and lower bounds on their *edge density*, i.e., the maximum number of edges with respect to the number of vertices. We study these questions in the general case and in two restricted variants that have been commonly studied for other classes of nearly-planar graphs. Namely, in an *outer-1-fbp* drawing all the vertices are incident to the unbounded face of the drawing, while in a *2-layer* 1-fbp drawing the graph

is bipartite, and the vertices of the two partitions lie on two parallel lines and the edges lie completely between these lines.

Our Contribution. In Sect. 2, we study inclusion relationships between the classes of 1- and 2-sided 1-fbp graphs and other classes of nearly-planar graphs. In Sect. 3, we present bounds on the edge density of these classes; see Table 1. We then prove in Sect. 4 that the recognition problem is NP-complete in general for both 1- and 2-sided models, while in Sect. 5 we present linear-time recognition and drawing algorithms for biconnected and maximal 2-layer 1-fbp graphs, and triconnected outer-1-fbp graphs in the 1-sided model. We conclude in Sect. 6 with open problems.

Table 1. Lower bounds (LB) and upper bounds (UB) on the edge-density

Model	2-layer			Outer			General		
	LB	UB	Ref.	LB	UB	Ref.	LB	UB	Ref.
1-sided	$\frac{5n-7}{3}$	$\frac{5n-7}{3}$	Theorem 4	$\frac{8n-13}{3}$	$\frac{8n-13}{3}$	Theorem 3	$\frac{13n-26}{3}$	$\frac{13n-26}{3}$	Theorem 2
2-sided	$2n - 4$	$3n - 7$	Theorem 7	$4n - 9$	$4n - 9$	Theorem 6	$6n - 18$	$8.6n - 15.6$	Theorems 5 and 8

Related Work. Apart from 1-planar [22], quasi-planar [2], and fan-planar [20] graphs, which have already been discussed, several other classes of nearly-planar graphs have been proposed over the last few years, e.g.: (i) *k-planar* [21], which generalize 1-planar graphs, as they can be drawn so that each edge is crossed at most *k* times; (ii) *fan-crossing free* [11], which complement fan-planar graphs, as they forbid fan crossings but allow each edge to cross several pairwise independent edges; and (iii) *RAC* [13], which admit straight-line drawings in which edges cross at right angles.

These classes have been mainly studied with respect to their edge density [1, 11, 13, 20–22] and to the complexity of their corresponding recognition problem, which has been proven NP-complete for most of the classes [4, 8, 17], except for quasi-planar and fan-crossing free graphs, whose complexities are unknown. However, for the restricted outer and 2-layer cases, several polynomial-time algorithms have been given [5, 6, 16, 19].

Fink et al. [14] considered a different style of edge bundling, where groups of locally parallel edges are bundled and only bundled crossings are allowed. Confluent drawings do not explicitly bundle edges, but represent edges by planar curves that are not interior-disjoint, so the parts that are used by several edges can be interpreted as bundles [12].

Preliminaries. A graph *G* admitting a 1-sided (2-sided) 1-fbp drawing is called 1-sided (2-sided) 1-fbp. Graph *G* is *maximal* if the addition of any edge destroys its 1-fan-bundle-planarity (in every drawing). Analogously, we define the (maximal) 1- or 2-sided *outer-1-fbp* and *2-layer 1-fbp* graphs. The drawings we consider are *almost simple*, meaning that no two fan-bundles of the same vertex cross. However, two edges incident to the same vertex may cross; refer to [3] for more

details A *rotation system* describes the clockwise order of the edges around each vertex of G .

A vertex u can be incident to several fan-bundles. Let B_u be such a fan-bundle. We say that B_u is *anchored at u* , which is the *origin* of B_u . We refer to the endpoint of B_u different from u (the point where all the edges of B_u are separated from each other) as the *terminal* of B_u , and to the endvertex different from u of any edge in B_u as a *tip* of B_u . A B_u - B_v -*following curve* is a curve that starts at u , follows B_u up to the crossing point with B_v , then follows B_v , and ends at v without crossing fan-bundles.

2 Relationships with Other Graph Classes

In this section, we discuss inclusion relationships between the classes of 1- and 2-sided 1-fbp graphs and other relevant classes of nearly-planar graphs; see Fig. 2a.

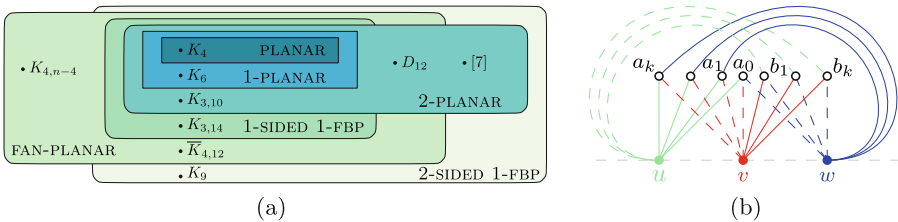


Fig. 2. (a) Inclusion relationships: The graph denoted by $\overline{K}_{4,12}$ is obtained from $K_{4,12}$ by joining on a path the 4 vertices of its first bipartition and on a second path the 12 vertices of its second one (see Fig. 2(a) in [20]). D_{12} corresponds to the graph obtained from the dodecahedron by adding a pentagram in each of its faces (see Fig. 2(b) in [20]). (b) A k -planar drawing of $K_{3,2k+1}$.

The inclusion relationship $1\text{-PLANAR} \subseteq 1\text{-SIDED 1-FBP} \subseteq \text{FAN-PLANAR}$ follows directly from the definition of 1-sided 1-fbp graphs, and the same holds for the inclusion $2\text{-PLANAR} \subseteq 2\text{-SIDED 1-FBP}$. Also, Binucci et al. [8] proved that the class of 2-planar graphs is incomparable with the one of fan-planar graphs.

The graph D_{12} obtained from the dodecahedron by adding a pentagram in each of its faces is 2-planar, fan-planar, and meets the common maximum density of these classes of graphs [20]. As we will see in Sect. 3, this graph is too dense to be 1-sided 1-fbp (and hence 1-planar). Since K_9 has more than $5n - 10$ edges, it is neither fan-planar nor 2-planar. However, K_9 is 2-sided 1-fbp; we give an illustration in [3]. We do not know whether K_{10} is 2-sided 1-fbp or not, but we know that there exists a value n for which K_n is not 2-sided 1-fbp, since these graphs have $O(n)$ edges; see Sect. 3. We recall that K_{10} is quasi-planar [9].

In [3], we demonstrate that the graph $\overline{K}_{4,12}$ obtained from $K_{4,12}$ by joining on a path the 4 vertices of its first bipartition and on a second path the 12 vertices of its second bipartition is 2-sided 1-fbp. This graph is fan-planar [20],

but not 2-planar (as it contains $K_{3,11}$ as a subgraph, which is not 2-planar; see Lemma 1). In addition, this particular graph contains 62 edges and is therefore too dense to be 1-sided 1-fbp (see Sect. 3).

We now show that $K_{3,11}$ is not 2-planar; note that even $K_{3,14}$ is 1-sided 1-fbp; refer to [3]. We also show that $K_{3,10}$ is 2-planar, by a more general proof.

Lemma 1. *For $k \geq 0$, graph $K_{3,4k+2}$ is k -planar, while $K_{3,4k+3}$ is not k -planar.*

Proof (sketch). To obtain a k -planar drawing of $K_{3,4k+2}$, we merge two copies of the drawing of $K_{3,2k+1}$ of Fig. 2b. For the negative result, we show that in any k -planar drawing of $K_{3,4k+3}$ there is an induced $K_{2,2}$ whose edges do not cross each other, and that the third vertex of the first bipartition can lie neither inside nor outside the region bounded by this $K_{2,2}$ (due to the crossing restrictions). For the full proof refer to [3]. □

As already noted, $K_{4,n-4}$ is fan-planar. We now show that there is a value n such that $K_{4,n-4}$ is not 2-sided 1-fbp. Hence, fan-planar (and also quasi-planar) graphs are not a subclass of 2-sided 1-fbp graphs. Note that $K_{4,14}$ is 2-sided 1-fbp; for an illustration refer to [3].

Theorem 1. *Graph $K_{4,n-4}$ is not 2-sided 1-fbp for $n \geq 571$.*

Proof (sketch). Assume that $K_{4,n-4}$ admits a 2-sided 1-fbp drawing Γ . Using Lemma 1, we can prove that in Γ there is a fan-bundle B_u anchored at a vertex u of the first bipartition that is shared by a certain number $z > 0$ of edges. We then consider the graph $K_{3,z}$ composed of the three vertices different from u in the first bipartition and of the z vertices of the second bipartition that are tips of B_u . Using Lemma 1 again, we can prove that in Γ there is a fan-bundle B_v anchored at another vertex v of the first bipartition that is shared by at least nine edges. Thus, B_u and B_v have at least nine common tips. Finally, we prove that this is not possible. For the full proof refer to [3]. □

3 Density

In this section, we consider Turán-type problems concerning 1-sided and 2-sided 1-fbp graphs, i.e., we ask what is the maximum number of edges they can have.

1-sided model. We start by giving a tight bound for the density.

Theorem 2. *A 1-sided 1-fbp graph with $n \geq 3$ vertices has at most $(13n - 26)/3$ edges, which is a tight bound.*

Proof. Let Γ be a 1-sided 1-fbp drawing of a *maximally dense* 1-sided 1-fbp graph G with n vertices, i.e., G has the largest possible number of edges. To estimate this number we transform G into a (not necessarily simple) maximal planar graph with no pair of *homotopic parallel edges*, i.e., both the interior and the exterior regions defined by any pair of parallel edges contain at least one vertex. Under this assumption, the maximum number of edges of a planar

multi-graph on n vertices is still $3n - 6$. We say that Γ contains an edge e , if there exists a drawn edge of G in Γ that is homotopic to e .

Consider two crossing fan-bundles B_u and B_v in Γ anchored at vertices u and v of G , respectively; see Fig. 3a. Let $(u, u_1), \dots, (u, u_\mu)$ and $(v, v_1), \dots, (v, v_\nu)$ be the edges bundled in B_u and B_v , in the order that they appear around the terminals of B_u and B_v in Γ , such that (u, u_1) and (v, v_1) are the edges that follow B_u and B_v along their terminals in clockwise direction. Note that B_u and B_v may share some tips, i.e., there may exist indices i, j , with $1 \leq i \leq \mu$ and $1 \leq j \leq \nu$, such that $u_i = v_j$ (e.g., by the maximality of G , $v_1 = u_\mu$ holds).

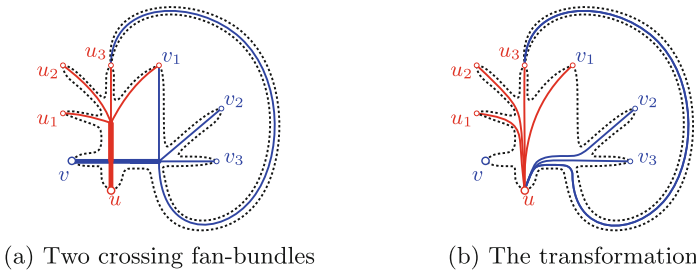


Fig. 3. The transformation used in Theorem 2 for the case $\mu = \nu = 4$. Note that $v_1 = u_\mu$ and $u_3 = v_\nu$. There exist two non-homotopic copies of (u, u_3) in (b)

Consider the edge (u, v) that one can draw in Γ as a B_u - B_v -following curve; we call (u, v) the *base-edge* of B_u and B_v . Since G is maximally dense, Γ contains this edge. For the same reason, Γ contains the edges $(v, u_1), \dots, (u_{\mu-1}, u_\mu), (u_\mu, v_1), \dots, (v_{\nu-1}, v_\nu)$, and (v_ν, u) that can be drawn by following either B_u , or B_v , or the unbundled parts of the edges incident to u and v (dotted in Fig. 3a).

We now describe a transformation of G ; see Fig. 3b. We remove from G all edges bundled in B_v and introduce edges $(u, v_2), \dots, (u, v_{\nu-1})$, drawn crossing-free completely in the interior of the region defined by edges $(u, v_1), \dots, (v_{\nu-1}, v_\nu)$, and (v_ν, u) in Γ . Note that this transformation eliminates the crossing between B_u and B_v , without introducing homotopic parallel edges. However, the transformed drawing has two edges less than Γ , namely (v, v_1) and (v, v_ν) . Applying this transformation recursively to every pair of crossing fan-bundles, we obtain a planar drawing Γ' of a (not necessarily simple) graph G' on the same vertices as G with no pair of homotopic parallel edges. Hence, G' has at most $3n - 6$ edges and $2n - 4$ faces. As noted above, G contains as many edges as G' plus twice the number of transformations. If a transformation involves exactly four vertices, then it introduces two faces of Γ' which will no be part of another transformation, as they are delimited by uncrossed edges, and Γ' has only one edge less than Γ . If a transformation involves at least five vertices, then it introduces at least three such faces of Γ' and Γ' has at most two edges less than Γ . Let f' be the number of faces of Γ' created by transformations that involve four vertices, and let f'' be the number of faces of Γ' created by the

remaining transformations. It follows that $f' + f'' \leq 2n - 4$. Thus, G has at most $3n - 6 + f'/2 + 2 \cdot \lfloor f''/3 \rfloor \leq 3n - 6 + 2 \cdot \lfloor (2n - 4)/3 \rfloor \leq (13n - 26)/3$ edges.

To show that this upper bound is tight, let \mathcal{P}_n be a planar graph on n vertices whose faces are of length five. By Euler's formula, \mathcal{P}_n has $(5n - 10)/3$ edges and $(2n - 4)/3$ faces. Since at each face of \mathcal{P}_n one can add four edges without violating 1-fan-bundle-planarity (see e.g. Fig. 1d), the resulting graph has $(5n - 10)/3 + 4(2n - 4)/3 = (13n - 26)/3$ edges, and the statement follows. \square

The same technique used in Theorem 2 can be applied to obtain tight bounds also in the outer and in the 2-layer models. The full proofs are in [3].

Theorem 3. *A 1-sided outer-1-fbp graph with $n \geq 5$ vertices has at most $(8n - 13)/3$ edges, which is a tight bound.*

Theorem 4. *A 1-sided 2-layer 1-fbp graph with $n \geq 5$ vertices has at most $(5n - 7)/3$ edges, which is a tight bound.*

2-sided model. We first establish a tight bound for outer-1-fbp graphs, and then upper and lower bounds for 2-layer and general 1-fbp graphs.

We start by presenting 2-sided outer-1-fbp graphs with n vertices and $4n - 9$ edges. A *flower drawing* of a graph is a 2-sided outer-1-fbp drawing in which (i) the vertices v_1, \dots, v_n lie on a circle \mathcal{C} in this clockwise order, (ii) each vertex v_i has exactly two fan-bundles, called *right* and *left* as seen from the center of \mathcal{C} , and (iii) for each $i = 1, \dots, n$, the right fan-bundle of v_i crosses the left fan-bundle of v_{i+1} , where $n + 1 = 1$; see Fig. 4a.

A *water lily* is a flower drawing of a graph with $n \geq 9$ vertices where the terminals of the fan-bundles are partitioned into three sets S_1, S_2 , and S_3 , such that (i) each set S_j , for $j = 1, 2, 3$, contains at least seven consecutive terminals, (ii) each two sets S_j and S_k , with $j \neq k$, have one terminal in common, which belongs to the right fan-bundle of a vertex, (iii) the terminal of the right fan-bundle of each vertex v_i is connected to the terminals of the left fan-bundles of vertices v_{i+1} and v_{i+2} , and (iv) the terminals in each set S_j , with $1 \leq j \leq 3$, are connected by a *zigzag-pattern* such that all but two faces have degree 3, the other two have degree 4 in order to avoid parallel edges; see Fig. 4a.

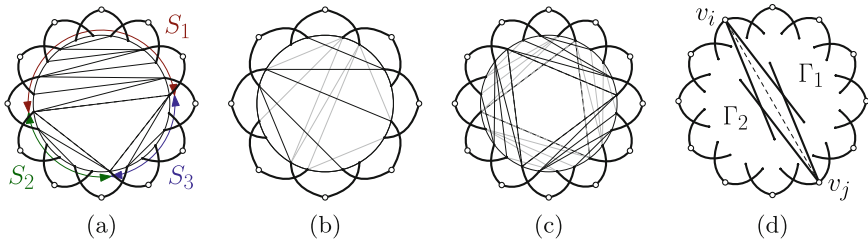


Fig. 4. (a) A water lily, (b) a 2-sided 1-fbp drawing of K_8 , (c) a 2-sided 1-fbp drawing of a graph with n vertices and $6n - 18$ edges for $n = 12$, and (d) crossing middle fan-bundles. In (b) and (c) the gray edges can be drawn on the outer face of the drawing by using twice as many fan-bundles.

Lemma 2. *A water lily drawing of a graph with $n \geq 9$ vertices has $4n - 9$ edges.*

Proof. Consider the graph H whose vertices are the terminals of the fan-bundles and whose edges are the unbundled parts of the edges of the water lily (non-bold in Fig. 4a). Graph H has $2n$ vertices, as each original vertex has one left and one right fan-bundle. By construction, H is biconnected and outerplanar. Also, all internal faces of H are triangular, except for six faces (two for each set S_j), which have size 4. Since an internally triangulated biconnected outerplanar graph on k vertices has $2k - 3$ edges, graph H has $2 \cdot 2n - 3 - 6 = 4n - 9$ edges. \square

The next theorem follows from the fact that we can draw on the outer face of a water lily another set of $2n - 9$ edges and obtain a 2-sided 1-fbp drawing with $6n - 18$ edges. Figure 4c shows that this can be done avoiding parallel edges; For the full proof refer to [3].

Theorem 5. *There are 2-sided 1-fbp graphs with n vertices and $6n - 18$ edges.*

We now show that 2-sided outer-1-fbp graphs are not denser than water lilies.

Theorem 6. *A 2-sided outer-1-fbp graph with $n \geq 3$ vertices has at most $4n - 9$ edges, which is a tight bound.*

Proof. The proof of the upper bound is by induction on n . For the base, observe that all graphs with $n \leq 6$ vertices have at most $4n - 9$ edges and that K_6 is 2-sided outer-1-fbp; see Fig. 1c. For the inductive step, let Γ be a 2-sided outer-1-fbp drawing of a graph G with $n \geq 7$. We show that G has at most $4n - 9$ edges. W.l.o.g., we assume that G has no vertex of degree less than 5, as otherwise we could remove it and apply induction.

Let v_1, \dots, v_n be the vertices of G in clockwise order on the outer face of Γ . We call *right* and *left bundle* of a vertex v_i the first and last fan-bundle in clockwise order around v_i , respectively, starting from the outer face. We assume that the left bundle of v_i crosses the right bundle of v_{i-1} ; otherwise, we could add two crossing dummy bundles.

First suppose that two middle bundles of two vertices v_i and v_j cross; see Fig. 4d. If $j = i + 1$ then the right bundle of v_i and the left bundle of v_j would be isolated and could be removed. So, we assume v_i and v_j are not consecutive and that (v_i, v_j) belongs to G , as otherwise we could add it. Hence, there is another pair of crossing bundles on the other side of (v_i, v_j) , as otherwise we could add two crossing dummy bundles. Thus, edge (v_i, v_j) splits Γ into two parts Γ_1 and Γ_2 containing n_1 and n_2 vertices. Since Γ_1 and Γ_2 contain v_i and v_j , we have $n_1 + n_2 = n + 2$. By induction, Γ_1 and Γ_2 have at most $4n_1 - 9$ and $4n_2 - 9$ edges. Hence, Γ has at most $(4n_1 - 9) + (4n_2 - 9) < 4n - 9$ edges.

Suppose now that no two middle bundles cross. W.l.o.g., we assume that each vertex is incident to at most one middle bundle, as otherwise we could merge them all into one. Let k be the number of vertices having a middle bundle. Let H be the graph whose vertices are the terminals of all fan-bundles and whose edges are the unbundled parts of the edges in Γ ; see Fig. 4a. Graph H is outerplanar, has $2n + k$ vertices, and thus at most $4n + 2k - 3$ edges ($2n + k$ outer edges, i.e.,

those on the outer face, and $2n + k - 3$ inner edges). The next two claims imply that H (and thus G) has at most $4n - 9$ edges.

Claim 1. *Graph H has at most $2n - k$ outer edges.*

Claim 2. *Graph H has at most $2n + k - 9$ inner edges.*

Proof. To prove Claim 1 note that, for each of the k vertices v_i with a middle bundle, the terminal of the right bundle of v_{i-1} lies between the terminals of the left and of the middle bundles of v_i along the outer face of H . So, there exist two outer edges of H representing the same edge (v_i, v_{i-1}) of G , one of which has to be removed to obtain simplicity. The claim follows by applying the same argument on the left bundle of v_{i+1} .

The proof of Claim 2 is based on the fact that inner edges connecting vertices that are at small distance along the outer face of H represent in G either self-loops or edges that are already represented by outer edges of H ; refer to [3].

The fact that the bound is tight follows from Lemma 2. □

The upper bound of the following theorem follows from Theorem 6. The lower bound exploits a construction similar to the one of Lemma 2; refer to [3].

Theorem 7. *A 2-sided 2-layer 1-fbp graph with $n \geq 3$ vertices has at most $3n - 7$ edges. There are 2-sided 2-layer 1-fbp graphs with n vertices and $2n - 4$ edges.*

We now give an upper bound on the edge density in the general case.

Theorem 8. *An n -vertex 2-sided 1-fbp graph has at most $(43n - 78)/5$ edges.*

Proof (sketch). Let Γ be a 2-sided 1-fbp drawing of a maximally dense graph G that has a maximum number of uncrossed edges. We define a planar graph G_p on the same vertex set as G that contains all the uncrossed edges of G in Γ . Since Γ has a maximum number of uncrossed edges, for each pair of crossing fan-bundles B_u and B_v , the base edge (u, v) of B_u and B_v is contained in G_p (note that multiple copies of (u, v) in Γ are pairwise non-homotopic). Hence, G_p has at most $3n - 6$ edges.

Next, we define another planar graph G'_p containing the vertices of G and the terminals of the fan-bundles of Γ , which we call *terminal vertices*. Graph G'_p has an edge for each base edge of Γ . Also, for each pair of crossing fan-bundles B_u and B_v with terminals t_u and t_v , graph G'_p contains (u, t_v) , (t_v, t_u) , (t_u, v) , and either (u, t_u) or (v, t_u) ; we call these edges *bridging edges*. Finally, for each unbundled part of each edge in Γ , graph G'_p has an edge connecting the corresponding terminal vertices of G'_p . Since G'_p is planar, it has at most $3(n + t) - 6$ edges, where t is the number of terminal vertices.

Note that G has as many edges as those in G'_p connecting terminal vertices, while each bridging edge has an endpoint that is not a terminal vertex. Since every two terminal vertices define four bridging edges, graph G has at most $3(n + t) - 6 - 2t = 3n + t - 6$ edges. For each edge e of G_p , there exist at most two adjacent crossing fan-bundle pairs, i.e., one on each side of e in Γ . So, $t \leq 4 \cdot (3n - 6) = 12n - 24$, which implies that G has at most $15n - 30$ edges. In [3], we improve this bound to $(43n - 78)/5$. □

4 NP-Completeness

In this section, we study the problem of deciding whether a graph G with a given rotation system R admits a 1-sided or a 2-sided 1-fbp drawing preserving R .

Theorem 9. *Given a graph G and a fixed rotation system R , it is NP-complete to decide whether G admits a 1-sided or 2-sided 1-fbp drawing preserving R .*

Proof (sketch). Membership in NP can be proved as for the fan-planarity [6, 8]. We prove NP-hardness by a reduction from 3-PARTITION [15] similar to the one of Bekos et al. [6] for the fan-planarity problem. An instance $\langle A, B \rangle$ of 3-PARTITION consists of an integer B and a set $A = \{a_1, \dots, a_{3m}\}$ of $3m$ positive integers in $(\frac{B}{4}, \frac{B}{2})$ such that $\sum_{i=1}^{3m} a_i = mB$. The problem asks whether A can be partitioned into m subsets, each of cardinality 3, such that the sum of the elements in each subset is B .

Central in the reduction of [6] is the *barrier gadget*, i.e., a subgraph whose edges cannot be crossed by other edges. This gadget is used to construct a *wall* surrounding the construction and a set of *obstacles* in its interior (gray in Fig. 5a). The edges between the obstacles (and the wall) constrain the routes of some paths, called *transversal* (bold in Fig. 5a), such that $\langle A, B \rangle$ has a solution if and only if all transversal paths can be routed without violating fan-planarity (in our case 1-fan-bundle-planarity).

For the 1-sided model, the connection between the two problems is the following. Each of the $3m$ columns in the interior of the wall consists of several sets of edges, called *cells*; one of these is *sparse* and contains as many edges as one of the elements in A ; the other ones are *dense* and contain significantly more edges. The length of the transversal paths ensures that each of them can cross $m - 3$ dense and 3 sparse cells. Since there are m such paths, a routing through the sparse cells implies a valid solution of $\langle A, B \rangle$, and vice versa.

In our case, it suffices to adjust the barrier gadget so that it is not traversable by any of the transversal paths without violating 1-fan-bundle-planarity. To this end, we propose the barrier gadget of Fig. 5b. The rest of the proof is similar to the one of [6]; details are given in [3].

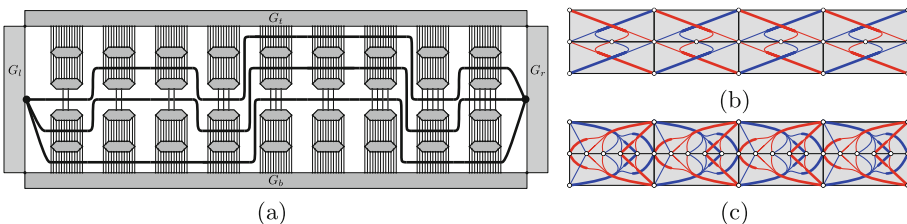


Fig. 5. (a) Sketch of the reduction with $m = 3$, $A = \{2, 2, 2, 3, 3, 3, 4, 5, 6\}$ and $B = 10$. The transversal paths are routed according to the following solution of 3-PARTITION: $A_1 = \{2, 3, 5\}$, $A_2 = \{2, 2, 6\}$ and $A_3 = \{3, 3, 4\}$. The barrier gadget in the (b) 1- and (c) 2-sided models, resp.

The proof under the 2-sided model requires the following modifications. Since each edge of the transversal paths can be crossed twice, we double the number of edges in each cell. Also, to avoid that transversal paths cross the same cell, we make consecutive pairs of edges in each cell cross; see Fig. 6. Finally, we modify the barrier gadget as in Fig. 5c. This concludes our proof. \square

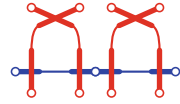


Fig. 6. Edges in the same cell.

5 Recognition and Drawing Algorithms

In this section, we give characterizations, recognition and drawing algorithms for subclasses of 1-sided 2-layer and outer 1-fbp graphs.

2-layer. Our results for 1-sided 2-layer 1-fan-bundle-planarity build upon concepts of Binucci et al. [7] for fan-planar graphs, who showed that a biconnected bipartite graph is maximal 2-layer fan-planar if and only if it is a *snake*, i.e., a chain of graphs $K_{2,h_i}, h_i \geq 2$, so that consecutive graphs share a pair of *merged* vertices, and no vertex is shared by more than two graphs. Also, it is 2-layer fan-planar if and only if it is a spanning subgraph of a snake [7]. Hence, a biconnected 2-layer 1-fbp graph is also a spanning subgraph of a snake. However, not every snake is 1-sided 2-layer 1-fbp, since $K_{2,4}$ is not 1-sided 2-layer 1-fbp (refer to [3] for more details); note that $K_{2,3}$ is (see Fig. 7a).

This leads to the following characterization, where a snake is a *baby snake* if each graph in the chain is a $K_{2,2}$ or a $K_{2,3}$. Hence, with the algorithm of Binucci et al. [7], we can also recognize and draw these graphs; see Theorem 10.

Theorem 10. *A biconnected graph is 2-layer 1-fbp if and only if it is a spanning subgraph of a baby snake; these can be recognized and drawn in linear time.*

We now relax biconnectivity and require maximality. It is known that a graph is maximal 2-layer fan-planar if and only if it is a *stegosaurus*, i.e., a chain of snakes that are connected at a *common cutvertex*, where each common cutvertex is incident to exactly two snakes, plus a set of vertices of degree 1 (*legs*) connected to the common cutvertices [7]. A stegosaurus is a *baby stegosaurus* if it consists of baby snakes and has no legs. A baby stegosaurus can be drawn 1-sided 2-layer 1-fbp by drawing its snakes and connecting them via their common cutvertices. The main argument is that no vertex incident to a $K_{2,2}$ has a leg; refer to [3] for more details.

Theorem 11. *Maximal 1-sided 2-layer 1-fbp graphs can be recognized and drawn in linear time.*

A leg not adjacent to a $K_{2,2}$ is a *big leg*. This yields the following characterization for non-maximal graphs, but not an efficient recognition algorithm.

Theorem 12. *A graph is 1-sided 2-layer 1-fbp if and only if it is a spanning subgraph of a baby stegosaurus with big legs.*

Outer. We give a linear-time algorithm for recognizing and drawing triconnected 1-sided outer-1-fbp graphs. We first describe properties of maximal biconnected and of triconnected 1-sided outer-1-fbp graphs; see Figs. 7b–c and [3].

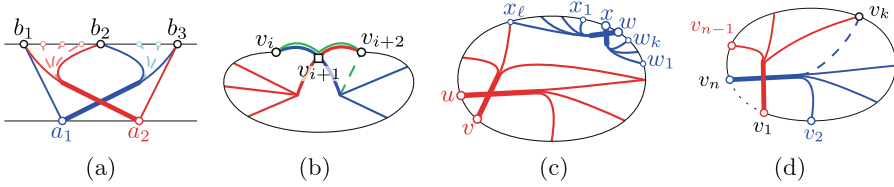


Fig. 7. (a) The only drawing of $K_{2,3}$ (solid) and the possible edges that can be added (dashed). Illustrations for (b) Lemma 3, (c) Lemma 4, and (d) Lemma 5.

Lemma 3. Any maximal biconnected 1-sided outer-1-fbp graph G has a 1-sided outer-1-fbp drawing in which all edges on the outer face are planar.

Lemma 4. In a 1-sided outer-1-fbp drawing Γ of a triconnected graph in which all edges on the outer face f are planar, (i) no inner edge is planar, (ii) the origins of two crossing fan-bundles are consecutive on f , and (iii) there is at most one crossing. Such a drawing is called a canonical drawing.

Lemma 5. A triconnected graph G with $n \geq 5$ vertices is 1-sided outer-1-fbp if and only if it consists of (i) a Hamiltonian path v_1, \dots, v_n , (ii) edges (v_n, v_i) and (v_1, v_j) , with $2 \leq i < k \leq j \leq n - 1$ for some k , (iii) edge (v_1, v_n) if $k \in \{2, n - 1\}$, and (iv) possibly edges (v_n, v_k) and (v_1, v_n) .

Proof (sketch). For the sufficiency, see Fig. 7d. For the necessity, if G is maximal, by Lemma 3, it has a 1-sided outer-1-fbp drawing Γ whose outer face is a simple planar cycle v_1, \dots, v_n, v_1 , so (i) holds. By Lemma 4, Γ is canonical, so there are only two fan-bundles B_{v_1} and B_{v_n} incident to every inner edge. Due to min-degree 3, all edges of (ii) and (v_n, v_k) exist. If $k \in \{2, n - 1\}$, then (v_1, v_n) exists for triconnectivity, so (iii) holds. If G is not maximal, then only (v_n, v_k) and (v_1, v_n) , if $k \in [3, n - 2]$, are not needed for triconnectivity, so (iv) holds. \square

From Lemma 5, we derive a linear-time recognition and drawing algorithm, since for our graphs a Hamiltonian path can be found efficiently; refer to [3].

Theorem 13. Triconnected 1-sided outer-1-fbp graphs can be recognized and drawn in linear time.

6 Conclusions

Our work opens several research directions: (i) Find recognition algorithms for 1- or 2-sided (biconnected) outer- or 2-layer 1-fbp graphs, (ii) close the gaps in the bounds of Table 1, (iii) discuss relationships with other nearly-planar graph classes, (iv) study k -fbp graphs, in which each fan-bundle is crossed at most k times, and (v) other models of edge bundling suitable for theoretical analyses.

References

1. Ackerman, E., Tardos, G.: On the maximum number of edges in quasi-planar graphs. *J. Comb. Theory Ser. A* **114**(3), 563–571 (2007)
2. Agarwal, P.K., Aronov, B., Pach, J., Pollack, R., Sharir, M.: Quasi-planar graphs have a linear number of edges. *Combinatorica* **17**(1), 1–9 (1997)
3. Angelini, P., Bekos, M.A., Kaufmann, M., Kindermann, P., Schneck, T.: 1-fan-bundle-planar drawings of graphs. CoRR [arXiv:1702.06163](https://arxiv.org/abs/1702.06163) (2017)
4. Argyriou, E.N., Bekos, M.A., Symvonis, A.: The straight-line RAC drawing problem is NP-hard. *J. Graph Algorithms Appl.* **16**(2), 569–597 (2012)
5. Auer, C., Bachmaier, C., Brandenburg, F.J., Gleißner, A., Hanauer, K., Neuwirth, D., Reislhuber, J.: Outer 1-planar graphs. *Algorithmica* **74**(4), 1293–1320 (2016)
6. Bekos, M.A., Cornelsen, S., Grilli, L., Hong, S.-H., Kaufmann, M.: On the recognition of fan-planar and maximal outer-fan-planar graphs. In: Duncan, C., Symvonis, A. (eds.) GD 2014. LNCS, vol. 8871, pp. 198–209. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45803-7_17
7. Binucci, C., Chimani, M., Didimo, W., Gronemann, M., Klein, K., Kratochvíl, J., Montecchiani, F., Tollis, I.G.: Algorithms and characterizations for 2-layer fan-planarity: from caterpillar to stegosaurus. *J. Graph Algorithms Appl.* **21**(1), 81–102 (2017)
8. Binucci, C., Giacomo, E.D., Didimo, W., Montecchiani, F., Patrignani, M., Symvonis, A., Tollis, I.G.: Fan-planarity: properties and complexity. *Theor. Comput. Syst.* **589**, 76–86 (2015)
9. Brandenburg, F.J.: A simple quasi-planar drawing of K_{10} . In: Hu, Y., Nöllenburg, M. (eds.) GD 2016. LNCS, vol. 9801, pp. 603–604. Springer, Cham (2016). <https://doi.org/10.1007/978-3-319-50106-2>
10. Buchin, K., Speckmann, B., Verbeek, K.: Flow map layout via spiral trees. *IEEE Trans. Vis. Comput. Graph.* **17**(12), 2536–2544 (2011)
11. Cheong, O., Har-Peled, S., Kim, H., Kim, H.: On the number of edges of fan-crossing free graphs. *Algorithmica* **73**(4), 673–695 (2015)
12. Dickerson, M., Eppstein, D., Goodrich, M.T., Meng, J.Y.: Confluent drawings: visualizing non-planar diagrams in a planar way. In: Liotta, G. (ed.) GD 2003. LNCS, vol. 2912, pp. 1–12. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24595-7_1
13. Didimo, W., Eades, P., Liotta, G.: Drawing graphs with right angle crossings. *Theor. Comput. Syst.* **412**(39), 5156–5166 (2011)
14. Fink, M., Hershberger, J., Suri, S., Verbeek, K.: Bundled crossings in embedded graphs. In: Kranakis, E., Navarro, G., Chávez, E. (eds.) LATIN 2016. LNCS, vol. 9644, pp. 454–468. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49529-2_34
15. Garey, M., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York (1979)
16. Giacomo, E.D., Didimo, W., Eades, P., Liotta, G.: 2-layer right angle crossing drawings. *Algorithmica* **68**(4), 954–997 (2014)
17. Grigoriev, A., Bodlaender, H.L.: Algorithms for graphs embeddable with few crossings per edge. *Algorithmica* **49**(1), 1–11 (2007)
18. Holten, D.: Hierarchical edge bundles: visualization of adjacency relations in hierarchical data. *IEEE Trans. Vis. Comput. Graph.* **12**(5), 741–748 (2006)
19. Hong, S., Eades, P., Katoh, N., Liotta, G., Schweitzer, P., Suzuki, Y.: A linear-time algorithm for testing outer-1-planarity. *Algorithmica* **72**(4), 1033–1054 (2015)

20. Kaufmann, M., Ueckerdt, T.: The density of fan-planar graphs. [arXiv:1403.6184](https://arxiv.org/abs/1403.6184) (2014)
21. Pach, J., Tóth, G.: Graphs drawn with few crossings per edge. *Combinatorica* **17**(3), 427–439 (1997)
22. Ringel, G.: Ein Sechsfarbenproblem auf der Kugel. *Abh. Math. Sem. Univ. Hamb.* **29**, 107–117 (1965)
23. Telea, A., Ersoy, O.: Image-based edge bundles: simplified visualization of large graphs. *Comput. Graph. Forum* **29**(3), 843–852 (2010)
24. Zhou, H., Xu, P., Yuan, X., Qu, H.: Edge bundling in information visualization. *Tsinghua Sci. Technol.* **18**(2), 145–156 (2013)
25. Zhou, H., Yuan, X., Qu, H., Cui, W., Chen, B.: Visual clustering in parallel coordinates. *Comput. Graph. Forum* **27**(3), 1047–1054 (2008)

Gap-Planar Graphs

Sang Won Bae¹, Jean-Francois Baffier², Jinhee Chun³, Peter Eades⁴,
Kord Eickmeyer⁵, Luca Grilli⁶, Seok-Hee Hong⁴, Matias Korman³,
Fabrizio Montecchiani⁶(✉), Ignaz Rutter⁷, and Csaba D. Tóth⁸

¹ Kyonggi University, Suwon, South Korea
swbae@kgu.ac.kr

² National Institute of Informatics, Tokyo, Japan
jf.baffier@nii.ac.jp

³ Tohoku University, Sendai, Japan

{jinhee,mati}@dais.is.tohoku.ac.jp

⁴ University of Sydney, Sydney, Australia
peter.eades@sydney.edu.au, shhong@it.usyd.edu.au

⁵ TU Darmstadt, Darmstadt, Germany
eickmeyer@mathematik.tu-darmstadt.de

⁶ University of Perugia, Perugia, Italy
{luca.grilli,fabrizio.montecchiani}@unipg.it

⁷ TU Eindhoven, Eindhoven, The Netherlands
i.rutter@tue.nl

⁸ California State University Northridge, Los Angeles, USA
csaba.toth@csun.edu

Abstract. We introduce the family of *k-gap-planar graphs* for $k \geq 0$, i.e., graphs that have a drawing in which each crossing is assigned to one of the two involved edges and each edge is assigned at most k of its crossings. This definition finds motivation in edge casing, as a k -gap-planar graph can be drawn crossing-free after introducing at most k local gaps per edge. We obtain results on the maximum density, drawability of complete graphs, complexity of the recognition problem, and relationships with other families of beyond-planar graphs.

1 Introduction

“Beyond-planar graphs” are informally defined as nonplanar graphs that can be represented with some forbidden edge crossing patterns (see, e.g., [29, 30, 36]).

Research started at the NII Shonan Meeting “Algorithmics for Beyond Planar Graphs.” The authors thank the organizers, and Yota Otachi for useful discussions. Bae was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2015R1D1A1A01057220). Baffier was supported by JST-ERATO Grant Number JPMJER1201, Japan. Eades and Hong were partially supported by ARC DP160104148. Korman was partially supported by MEXT KAKENHI No. 15H02665, 17K12635 and JST ERATO Grant Number JPMJER1305. Tóth was supported in part by the NSF awards CCF-1422311 and CCF-1423615.

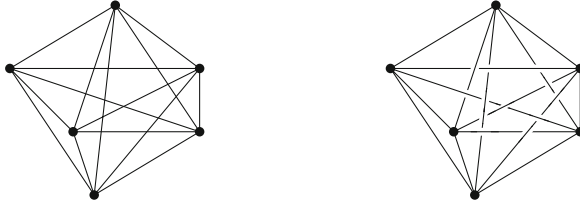


Fig. 1. (a) A drawing of a graph G and (b) its cased version where each edge is interrupted at most twice, i.e., a 2-gap-planar drawing of G .

Research on this topic is attracting increasing attention within the communities of graph theory, graph algorithms, graph drawing, and computational geometry, as these graphs represent a natural generalization of planar graphs, and their study can provide significant insights for the design of effective methods to visualize real-world networks. Indeed, the motivation for this line of research stems from both the interest raised by the combinatorial and geometric properties of these graphs, and experiments showing how the absence of particular edge crossing patterns has a positive impact on the readability of a graph drawing [31].

Among the investigated families of beyond-planar graphs are: *k-planar graphs* (see, e.g., [11, 34, 38]), which can be drawn with at most $k > 0$ crossings per edge; *k-quasiplanar graphs* (see, e.g., [2, 3, 22]), where there are no $k > 2$ pairwise crossing edges; *fan-planar graphs* (see, e.g., [9, 12, 32]), where no edge can be crossed by two independent edges; *fan-crossing-free graphs* [17], where crossings between an edge and two adjacent edges are forbidden; *planarly-connected graphs* [1], in which each pair of crossing edges is independent and there is a crossing-free edge that connects their endpoints; *RAC graphs* (refer, e.g., to [19]), which admit a straight-line (or polyline with few bends) drawing with right-angle crossings.

In this paper we introduce *k-gap-planar*. Intuitively speaking, each crossing is assigned to one of the two involved edges and each edge is assigned at most k crossings (see Sect. 2). This definition generalizes that of *k-planar graphs*, and it is practically motivated by *edge casing*, a method commonly used to alleviate the visual clutter generated by crossing lines in a diagram [5, 21]. In a *cased drawing* of a graph, each crossing is resolved by locally interrupting one of the two crossing edges. Clearly, minimizing the number of gaps per edge is one of the desirable goals in this situation, and a *k-gap-planar graph* can be equivalently defined as a graph that admits a cased drawing in which each edge has at most k gaps. Figure 1 shows a drawing of a graph and its version with edge casing. Eppstein et al. [21] studied many optimization problems related to edge casing, assuming the input to be a drawing (rather than a graph). In particular, the problem of minimizing the maximum number of gaps (called *tunnels*) for any edge of a drawing can be solved in polynomial time (see also Sect. 2). We also remark that a similar drawing paradigm is used by *partial edge drawings (PEDs)*, in which the central part of each edge is erased, while the two remaining stubs are required to be crossing-free (see, e.g., [15, 16]).

Our results can be summarized as follows:

- Every k -gap-planar graph with n vertices has $O(\sqrt{k} \cdot n)$ edges (Sect. 3). If $k = 1$, a bound of $5n - 10$ edges is proved for 1-gap-planar multigraphs, which is tight as there exist k -gap-planar (simple) graphs with these many edges. Note that this density bound equals that of 2-planar graphs [38].
- The complete graph K_n is 1-gap-planar if and only if $n \leq 8$ (Sect. 4).
- Deciding whether a graph is 1-gap-planar is NP-complete, even when the input graph comes with a fixed rotation system that is part of the input (Sect. 5). We remark that analogous recognition problems for other families of beyond-planar graphs are also NP-hard (see, e.g., [7, 9, 12, 13, 25, 35]), while polynomial algorithms are known only in restricted settings (see, e.g., [6, 9, 13, 18, 20, 27, 28]).
- We study relationships of the k -gap-planar family with other beyond-planar families. For all $k \geq 1$, the class of $2k$ -planar graphs is properly included in the class of k -gap-planar graphs, which in turn is properly included in the $(2k + 2)$ -quasiplanar graphs (Sect. 6). It is worth observing that recent papers proved that k -planar graphs are $(k + 1)$ -quasiplanar [4, 26].

For reasons of space some proofs and technicalities have been omitted and can be found in [8].

2 Preliminaries and Basic Results

A *drawing* Γ of a graph $G = (V, E)$ is a mapping of the vertices of V to distinct points of the plane, and of the edges of E to Jordan arcs connecting their corresponding endpoints but not passing through any other vertex. If two edges are incident to the same vertex, then they do not cross in Γ ; else, they have at most one common interior point where they cross transversely. For a subset $E' \subseteq E$, $\Gamma[E']$ denotes the restriction of Γ to the curves representing the edges of E' . Drawing Γ is *planar* if no edge is crossed. The *crossing number* $\text{cr}(G)$ of a graph G is the smallest number of edge crossings over all drawings of G . A graph is *planar* if it admits a planar drawing. A planar drawing subdivides the plane into topologically connected regions, called *faces*. The unbounded region is the *outer face*. A *planar embedding* of a planar graph G is an equivalence class of topologically equivalent drawings of G . A *plane graph* is a planar graph with a planar embedding. The *crossing graph* $C(\Gamma)$ of a drawing Γ is the graph having a vertex v_e for each edge e of G , and an edge (v_e, v_f) if and only if edges e and f cross in Γ . The *planarization* Γ^* of Γ is the plane graph formed from Γ by replacing each crossing by a *dummy vertex*. To avoid ambiguities, we call *real vertices* the vertices of Γ^* that are not dummy.

Let Γ be a drawing of a graph G . We shall assume that exactly two edges of G cross in one point p of Γ , and we say that these two edges are *responsible* for p . A *k-gap assignment* of Γ maps each crossing point of Γ to one of its two responsible edges so that each edge is assigned with at most k of its crossings; see, e.g., Fig. 1(b). The *gap* of an edge is the number of crossings assigned to it.

An edge with at least one gap is *gapped*, or a *gap edge*, else it is *gap free*. A drawing is *k-gap-planar* if it admits a *k-gap* assignment. A graph is *k-gap-planar* if it has a *k-gap-planar* drawing. Note that the 0-gap-planar graphs coincide with the planar graphs, and that *k-gap-planarity* is a monotone property: every subgraph of a *k-gap-planar* graph is *k-gap-planar*. From the pigeonhole principle we have:

Property 1. Let Γ be a *k-gap-planar* drawing of a graph $G = (V, E)$. For any $E' \subseteq E$, $\Gamma[E']$ contains at most $k \cdot |E'|$ crossings.

A *k-gap* assignment of a drawing Γ corresponds to orienting the edges of the crossing graph $C(\Gamma)$ such that each vertex has indegree at most k (intuitively, orienting a crossing towards an edge means we assign the crossing to that edge). Since finding a lowest indegree orientation of a graph corresponds to finding its pseudoarboricity [23, 39], Property 2 follows. A *pseudoforest* is a graph in which every connected component has at most one cycle, and the *pseudoarboricity* of a graph is the smallest number of pseudoforests needed to cover all its edges.

Property 2. A graph is *k-gap-planar* if and only if it admits a drawing whose crossing graph has pseudoarboricity at most k .

Given a drawing Γ of a graph $G = (V, E)$, finding the minimum k such that Γ is *k-gap-planar* can be solved in $O(|E|^4)$ time, due to the fact that finding a lowest indegree orientation of $C(\Gamma)$ can be solved in time quadratic in the number of edges of $C(\Gamma)$ [40].

3 Density of *k-gap-planar* Graphs

We begin with an upper bound on the number of edges of *k-gap-planar* graphs.

Theorem 1. *A k-gap-planar graph on $n \geq 3$ vertices has $O(\sqrt{k} \cdot n)$ edges.*

Proof. The crossing number of a graph G with n vertices and m edges is bounded by $\text{cr}(G) \geq \frac{1024}{31827} \cdot m^3/n^2$ when $m \geq \frac{103}{6}n$ [37]. Combined with the bound $\text{cr}(G) \leq k \cdot m$ (Property 1), we obtain

$$\frac{1024}{31827} \cdot \frac{m^3}{n^2} \leq \text{cr}(G) \leq km,$$

which implies $m \leq \max(5.58\sqrt{k}, 17.17) \cdot n$, as required. □

Better upper bounds are possible for small values of k , in particular for $k = 1$. Pach et al. [37] proved that a graph G with $n \geq 3$ vertices satisfies $\text{cr}(G) \geq \frac{7}{3}m - \frac{25}{3}(n - 2)$. Combined with the bound $\text{cr}(G) \leq k \cdot m$, we have

$$m \leq \frac{25(n - 2)}{7 - 3k}.$$

For $k = 1$ (i.e., for 1-gap-planar graphs), this gives $m \leq 6.25n - 12.5$. We now show how to improve this bound to $m \leq 5n - 10$. The idea is to follow a strategy developed by Pach and Tóth [38] and Bekos et al. [11] on the density of 2- and 3-planar graphs, with several important differences.

We start by stating the assumptions and notations for the proof of Theorem 2. In order to accommodate the elementary operations in the proof, we work on a broader class of graphs, namely multigraphs admitting a drawing without homotopic¹ parallel edges.

(i) For any $n \in \mathbb{N}$, $n \geq 3$, let $G = (V, E)$ be a 1-gap-planar multigraph with n vertices that has the maximum number of edges possible over all n -vertex 1-gap-planar multigraphs without homotopic parallel edges; (ii) let Γ be a 1-gap-planar drawing of G with the minimum number of edge crossings over all possible 1-gap-planar drawings of G with non-homotopic parallel edges; and (iii) let $H = (V, E')$ be a sub-multigraph of G , where $E' \subseteq E$ is a multiset of edges that are pairwise noncrossing in $\Gamma[E']$. (iv) We assume that over all choices of G , Γ , and H described above, the multigraph H is maximum and, in case of ties, has the fewest connected components.

Our proof is based on the next technical lemma.

Lemma 1. *The multigraph H is a triangulation, that is, a plane multi-graph in which every face is bounded by a walk with three vertices and three edges.*

We can now show that $|E| \leq 5n - 10$.

Theorem 2. *A 1-gap-planar graph on $n \geq 3$ vertices has at most $5n - 10$ edges.*

Proof. By Lemma 1, we know that $H = (V, E')$ is a triangulation. By Euler’s polyhedron theorem, it has $3n - 6$ edges and $2n - 4$ triangular faces. Consider the edges in $E'' = E \setminus E'$. It remains to show that $|E''| \leq 2n - 4$.

The embedding of edge $e \in E''$ is a Jordan arc that visits two or more triangle faces of H . We call the first and last triangles along e the *end triangles* of e . For an end triangle Δ , the connected component of $e \cap \Delta$ incident to a vertex of Δ is called an *end portion*. We use the following charging scheme.

Each edge $e \in E''$ charges one unit to a triangle face of H . If e has an end portion that has a gap neither in the interior nor on the boundary of the corresponding end triangle Δ , then e charges one unit to Δ . (If neither end portions of e has a gap in the interior or on the boundary of its end triangle, then e charges one arbitrary end triangle.) Otherwise the two end portions of e lie in two adjacent triangles, say, Δ_1 and Δ_2 , and e uses its gap to cross the common edge on the boundary between them; in this case e charges one unit to Δ_1 or Δ_2 as follows: If the gap of the common edge between Δ_1 and Δ_2 is used for an end portion of $e' \cap \Delta_1$ for another edge $e' \in E''$ and e' charges Δ_1 , then e charges Δ_2 , otherwise it charges Δ_1 .

We claim that each face of H receives at most one unit of charge. Let $\Delta = \triangle abc$ be a face in H . Note that if Δ receives positive charge from an edge $e \in E''$,

¹ Two parallel edges are *homotopic* if at least one of the two regions defined by these two edges contains no vertex in its interior.

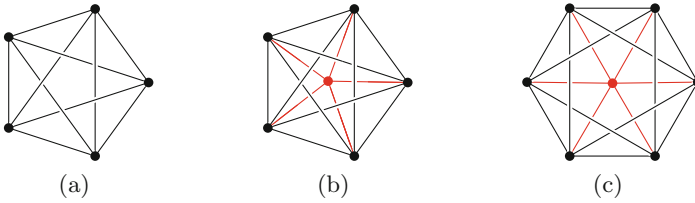


Fig. 2. Patterns that produce 1-gap-planar graphs with n vertices and $5n - \Theta(1)$ edges.

then an end portion of e lies in Δ , and does not use any gap in the interior of Δ . Consequently if Δ received positive charge from edges $e_1, e_2 \in E''$, then the end portions of e_1 and e_2 in Δ cannot cross, and they are incident to the same vertex of Δ . Therefore, all edges in E'' that charge Δ are incident to the same vertex of Δ , say a , and cross the edge of Δ opposite to a , namely (b, c) . Let $\Delta' = \Delta' bcd$ be the face of the plane graph H on the opposite side of (b, c) .

The gap of edge (b, c) can be used for at most one crossing along (b, c) . If the gap of (b, c) is used for a crossing with one of the end portions in Δ , then e sends 1 unit charge to Δ . The only other edge that could possibly send a charge to Δ is the edge $(a, d) \in E''$ that uses its own gap to cross (b, c) . However, in this case, (a, d) charges one unit to Δ' in our charging scheme. If the gap of (b, c) is not used for any of these end portions in Δ , then the edge (a, d) may send 1 unit charge to Δ . Overall, Δ receives at most 1 unit of charge. Consequently, $|E''|$ is bounded above by the number of faces of H , which is $2n - 4$, as required. \square

We now show that the bound of Theorem 2 is worst-case optimal. A 2-planar graph with n vertices and $5n - 10$ edges is also 1-gap-planar (see Lemma 6). Pach and Tóth [38] construct such a graph by starting with a plane graph with pentagonal faces (e.g., using nested copies of an icosahedron), and then add all five diagonals in each pentagonal face; see Fig. 2(a). This construction yields a 1-gap-planar graph with n vertices and $m = 5n - 10$ edges for all $n \geq 20, n \equiv 5 \pmod{15}$.

We can modify this construction by inserting a new vertex in one or more pentagons, and connecting it to the 5 vertices of the pentagon; see Fig. 2(b). Every new edge crosses exactly one diagonal of the pentagon, so the new crossings can be charged to the new edges. Since every new vertex has degree 5, the equation $m = 5n - 10$ prevails. By inserting a suitable number of vertices into pentagons, we obtain constructions for $n \in \mathbb{N}$ such that $20 \leq n \leq 32$ or $n \geq 38$. A similar construction is based on hexagonal faces; see Fig. 2(c). Start with a fullerene, that is, a 3-regular, plane graph G_0 with n_0 vertices, 12 pentagon faces, and $n_0/2 - 10$ hexagon faces (including the external face). Add diagonals in each face to connect a vertex to their second neighbors (the graph is 2-planar so far); finally insert a new vertex in each face of G_0 , and connect them to all vertices of that face. We obtain a 1-gap-planar graph G . The number of vertices is $n = n_0 + 12 + (n_0/2 - 10) = \frac{3}{2}n_0 + 2$, and the number of edges is $m = \frac{3}{2}n_0 + 10 \cdot 12 + 12 \cdot (n_0/2 - 10) = \frac{15}{2}n_0 = 5n - 10$. Fullerenes exist for

$n_0 = 20$ and for all even integers $n_0 \geq 24$ [14]. This yields a lower bound of $5n - 10$ for $n = 32$ and for all $n \geq 35$ where $n \equiv 2 \pmod 3$. However, similarly to the previous construction, the equation $m = 5n - 10$ prevails if we *delete* up to 12 vertices inserted into pentagons. Consequently, the upper bound $5n - 10$ in Theorem 2 is tight for all $n \geq 20$.

Theorem 3. *For every $n \geq 20$ there exists a 1-gap-planar graph G with n vertices and $5n - 10$ edges.*

4 1-gap-planar Drawings of Complete Graphs

Theorem 4. *The complete graph K_n is 1-gap-planar if and only if $n \leq 8$.*

Proof. Figure 3(a) shows a 1-gap-planar drawing of K_8 , and by monotonicity the graphs K_1, \dots, K_7 are 1-gap-planar as well. We now prove that K_9 is not 1-gap-planar, which again by monotonicity settles all cases K_n for $n \geq 9$.

Since K_9 has 36 edges and $cr(K_9) = 36$, a 1-gap-planar drawing of K_9 can only arise from assigning exactly one gap to each edge in a crossing-minimal drawing of K_9 (cf. Property 1). We obtain a contradiction by showing that in every crossing-minimal drawing of K_9 some edge has no crossing at all.

Let Γ^* be the planarization of such a crossing-minimal drawing Γ . Note that Γ^* has $n^* = 45$ vertices and $m^* = 108$ edges (since it has 9 real vertices of degree 8 and 36 dummy vertices of degree 4), so by Euler’s formula, the number of faces of Γ^* is $f^* = m^* - n^* + 2 = 108 - 45 + 2 = 65$. For a real vertex u of Γ^* , we denote by $F(u)$ the set of faces of Γ^* that are incident to u . We claim that Γ^* is biconnected and $|F(u)| = 8$ for every real vertex u of Γ^* .

Suppose, for a contradiction, that Γ^* is not biconnected. Then it contains a cut-vertex c , which is either a dummy or a real vertex. If c is a dummy vertex, note that it is adjacent to exactly two connected components of $\Gamma^* \setminus \{c\}$. Then we can reflect the drawing of one of the two components, thereby eliminating the crossing at c , which contradicts the crossing-minimality of Γ . We now show that no real vertex is a cut-vertex in Γ^* . Every 3-cycle in K_9 forms a simple cycle in Γ^* (since Γ is a simple drawing and thus adjacent edges do not cross). On the other hand, any three real vertices in Γ^* are part of a 3-cycle in K_9 , and thus part of a simple cycle in Γ^* . Hence, no real vertex is a cut-vertex in Γ^* . Finally, $|F(u)| = 8$ because every real vertex u has degree 8 and Γ^* is biconnected.

It follows that there are real vertices u, v which share a face (i.e. $F(u) \cap F(v) \neq \emptyset$), as otherwise there would have to be $\sum_u |F(u)| = 9 \cdot 8 = 72 > 65 = f^*$ faces. But now the edge (u, v) can be redrawn inside this face, and since Γ was assumed to be crossing-minimal this edge can not have had any crossing to begin with. □

5 Recognizing 1-gap-planar Graphs

We use 1GAPPLANARITY to denote the problem of deciding whether a given graph G is 1-gap-planar. We show that 1GAPPLANARITY is NP-complete, and

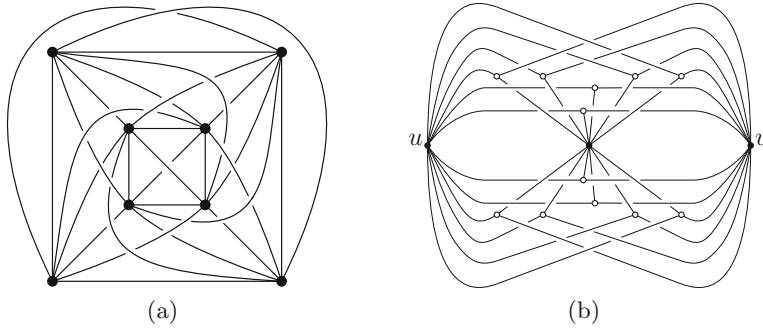


Fig. 3. A 1-gap-planar drawing of (a) K_8 and (b) $K_{3,12}$.

we use a reduction from 3PARTITION. Recall that an instance of 3PARTITION consists of a multiset $A = \{a_1, a_2, \dots, a_{3m}\}$ of $3m$ positive integers in the range $(B/4, B/2)$, where B is an integer such that $B = 1/m \cdot \sum_{i=1}^{3m} a_i$, and asks whether A can be partitioned into m subsets A_1, A_2, \dots, A_m , each of cardinality 3, such that the sum of integers in each subset is B . This problem is strongly NP-hard [24], and thus we may assume that B is bounded by a polynomial in m .

The fact that 1GAPPLANARITY is in NP can easily be shown by exploiting Property 2.

Our reduction is reminiscent to the reduction used in [9]. However, the proof in [9] holds only for the case in which a clockwise order of the edges around each vertex is part of the input, i.e., only if the *rotation system* of the input graph is fixed. A similar reduction is also used in [10], in which the rotation system assumption is not used. However, the gadgets used in [10] have a unique embedding. We do not use the fixed rotation system assumption, nor we can easily derive a unique embedding for our gadgets, and thus have to deal with additional challenges in our proof. In what follows we define a “blob” graph that will be used to enforce an ordering among the edges adjacent to certain vertices. Consider the complete bipartite graph $K_{3,12}$, whose crossing number is 30 [33,41]. We exhibit a 1-gap-planar drawing of $K_{3,12}$ with exactly 30 gaps in Fig. 3(b). Note that two degree-12 vertices, u and v , are drawn on the outer face. Since $K_{3,12}$ has 36 edges, the next lemma easily follows.

Lemma 2. *Every 1-gap-planar drawing of $K_{3,12}$ has at most 6 gap-free edges.*

A *blob* B is a copy of $K_{3,12}$. A *gapped chain* C of a 1-gap-planar drawing is a closed, possibly nonsimple, curve such that any point of C either belongs to a gap edge or it corresponds to a vertex.

Lemma 3. *Let u and v be any two degree-12 vertices of B . Every 1-gap-planar drawing Γ of B contains a gapped chain C containing u and v .*

Sketch of proof. Let Γ^* be the planarization of Γ . Let Γ' be the subgraph of Γ^* consisting only of those edges that correspond to or belong to gap edges of Γ .

We prove that Γ' contains two edge-disjoint paths from u to v . Note that these two edge-disjoint paths may meet at real vertices and at dummy vertices (i.e., a crossing between two gap edges). A curve that goes through these two paths is the desired gapped chain. According to Menger's theorem, two such paths exist if and only if every (u, v) -cut of Γ' has size at least 2, where a (u, v) -cut of Γ' is a set of edges of Γ' whose removal disconnects u and v . Such edge cuts correspond to cycles in the dual, which in turn correspond to curves that separate u and v by crossing a set of edges. By Lemma 2, one can show that any such curve crosses at least two gap edges in the original drawing Γ . \square

We are now ready to show how to transform an instance A of 3PARTITION into an instance G_A of 1GAPPLANARITY. We start by defining some gadgets for our construction. A *path gadget* P_k is a graph obtained by merging a sequence of $k > 0$ blobs as follows. Denote by B_1, B_2, \dots, B_k , k blobs such that u_i and v_i are two vertices of degree 12 in B_i . We let $v_i = u_{i+1}$ for $i = 1, \dots, k-1$, each of these vertices is an *attaching vertex*. Thus, P_k has $k + 1$ attaching vertices. A 1-gap-planar drawing of P_k is such that any two gapped chains of any two blobs B_i and B_j ($i < j$) do not share points, except at a possible common attaching vertex. A schematization of P_k (for $k = 3$) is shown in Fig. 4(a). A *top beam* G_t is a path gadget P_k with $k = 3m(\lceil B/2 \rceil + 2) + 1$. Recall that G_t has $3m(\lceil B/2 \rceil + 2) + 2$ attaching vertices. A *right wall* G_r is a path gadget P_k with $k = 2$. Symmetrically, a *bottom beam* G_b is a path gadget with $k = 3m(\lceil B/2 \rceil + 2) + 1$, and a *left wall* G_l is a path gadget with $k = 2$. A *global ring* R is obtained by merging G_t, G_r, G_b , and G_l in a cycle as in Fig. 4(b). Again, in any 1-gap-planar drawing Γ_R of R , the gapped chains of any two blobs B_i and B_j do not share points, except at a possible common attaching vertex. Thus, Γ_R contains a gapped chain C_R that is the union of all the gapped chains of the blobs of R .

We start the construction of G_A with a global ring R . Let $\alpha, \beta, \gamma, \delta$ be the attaching vertices shared by G_l and G_t, G_t and G_r, G_r and G_b, G_b and G_l , respectively (see also Fig. 4(b)). First we add the edges (α, β) and (γ, δ) . Denote as R^+ the resulting graph, and consider a 1-gap-planar drawing of this graph. The gapped chain of R subdivides the plane into a set of connected regions, such that only two of them contain all of α, β, γ , and δ on their boundaries. We denote these two regions as r_1 and r_2 . For ease of illustration, we assume that one of them is infinite (as in Fig. 4(b)), say r_2 . Since the drawing is 1-gap-planar, each of (α, β) and (γ, δ) is drawn entirely in one of these two regions. We assume that both these two edges are drawn in the same region, say r_2 , and we will later show that this is the only possibility in any 1-gap-planar drawing of the final graph G_A . We continue by connecting the top and bottom beams by a set of $3m$ *columns*; refer to Fig. 4(c). Each column consists of $2m - 1$ *cells*; a cell consists of a set of pairs of crossing edges, called its *vertical pairs*. In particular, there are $m - 1$ *bottom cells*, one *central cell* and $m - 1$ *top cells*. Cells of the same column are separated by $2m - 2$ path gadgets, called *floors*. Note that we are assuming a particular left-to-right order for the attaching vertex of a floor, we will see that this is the only possible order in a 1-gap-planar drawing. The central cells (we have $3m$ of them in total) have a number of vertical pairs depending on

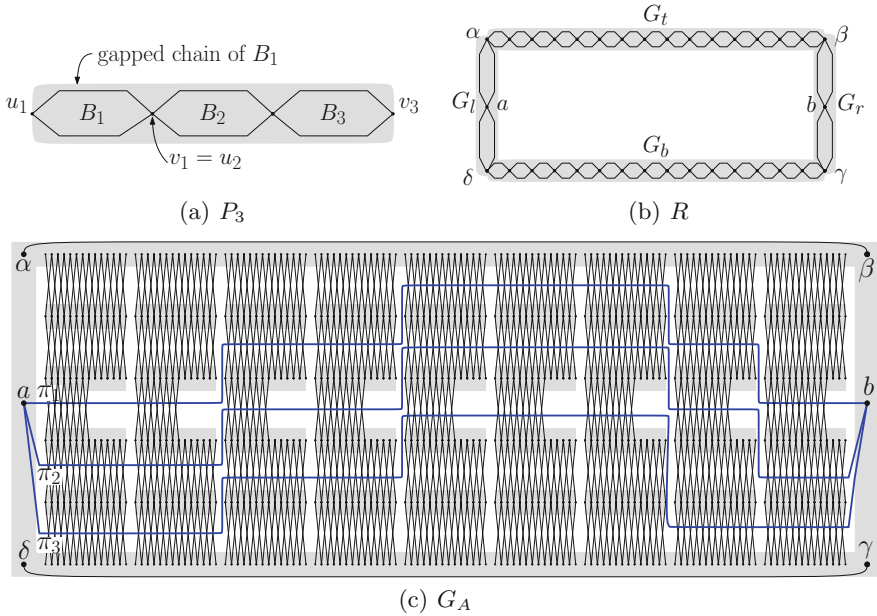


Fig. 4. (a) Schematization of a path gadget P_3 . (b) A global ring R . (c) Schematization of the instance G_A with $m = 3$, $A = \{7, 7, 7, 8, 8, 8, 8, 9, 10\}$ and $B = 24$. Transversal paths are routed according to the following solution of 3PARTITION $A_1 = \{7, 7, 10\}$, $A_2 = \{7, 8, 9\}$ and $A_3 = \{8, 8, 8\}$. For simplicity, the gapped chains of the various blobs are not shown, as well as vertex w and all the degree-2 vertices of the transversal paths.

the elements of A . Specifically, the central cell C_i of the i -th column contains a_i vertical pairs connecting its delimiting floors ($i \in \{1, 2, \dots, 3m\}$). Each of the remaining cells each has $\lceil B/2 \rceil + 1$ vertical pairs. Hence, a noncentral cell contains more edges than any central cell. Further, the number of attaching vertices of a floor can be computed based on how many vertical pairs must be connected to the gadget. It is now straightforward to see that it is not possible to draw both a column and one of (α, β) and (γ, δ) in r_1 or r_2 without violating 1-gap-planarity. Hence, we shall assume that both (α, β) and (γ, δ) are in r_2 and that all the columns are in r_1 . Consider now a 1-gap-planar drawing of a column. If we invert the left-to-right order of the attaching vertices of a floor (i.e., we mirror its drawing), then the resulting drawing is not 1-gap-planar, since each floor delimits at least one noncentral cell with $\lceil B/2 \rceil + 1$ vertical pairs. Moreover, since each vertical pair has a gap edge, two vertical pairs cannot cross each other in a 1-gap-planar drawing, and thus the drawings of the columns are disjoint one another. Finally, let a and b be the attaching vertices of the left and right walls distinct from $\alpha, \beta, \gamma,$ and δ . We connect a and b with m pairwise internally disjoint paths, called *transversal paths*; each transversal path has exactly $(3m - 3)(\lceil B/2 \rceil + 1) + B$ edges. The routing of these paths will

be used to determine a solution of A , if it exists. Thus, we aim at forcing the transversal paths to be inside r_1 in a 1-gap-planar drawing of the graph. For this purpose, adding a vertex w connected to all the attaching vertices of G_t and G_b will suffice. Due to the presence of the columns in r_1 , vertex w must be in r_2 and, due to the edges (α, β) and (δ, γ) in r_2 , all its incident edges (except at most two) are gapped. Thus, the transversal paths must be drawn in r_1 . This concludes the construction of G_A .

We can prove the following.

Theorem 5. *The 1GAPPLANARITY problem is NP-complete.*

We conclude by observing that our proof can be easily adjusted for the setting in which the rotation system of the input graph is fixed. We call this problem 1GAPPLANARITYWITHROTSYS. It suffices to choose a rotation system for G_A that guarantees the existence of a 1-gap-planar drawing ignoring the transversal paths (we already discussed the details of this drawing), and such that the transversal paths are attached to a and b with the ordering of their edges around a reversed with respect to the ordering around b . The membership of the problem to NP can be easily verified. Thus, the next theorem follows.

Theorem 6. *The 1GAPPLANARITYWITHROTSYS problem is NP-complete.*

6 Relationship Between k -gap-planar Graphs and Other Families of Beyond Planar Graphs

In this section we prove the following theorem.

Theorem 7. *For every integer $k \geq 1$, the following relationships hold.*

$$(2k)\text{-PLANAR} \subsetneq k\text{-GAP-PLANAR} \subsetneq (2k+2)\text{-QUASIPLANAR}$$

We begin by showing the following.

Lemma 4. *For all $k \geq 1$, every k -gap-planar drawing is $(2k+2)$ -quasiplanar.*

Proof. Recall that a graph G is q -quasiplanar, for $q \in \mathbb{N}$, if it admits a drawing in which there is no subset of q pairwise crossing edges, or equivalently if every subset of q edges has less than $\binom{q}{2} = q(q-1)/2$ crossings. On the other hand, in a k -gap-planar drawing there are at most kq crossings among any q edges (Property 1). Consequently, a k -gap-planar graph is $(2k+2)$ -quasiplanar. \square

We also need to show that for every $k \in \mathbb{N}$ there is a $(2k+2)$ -quasiplanar graph that is not k -gap-planar. We prove a stronger statement:

Lemma 5. *For all $k \geq 1$, there is a 3-quasiplanar graph G_k that is not k -gap-planar.*

Proof. Let $k \in \mathbb{N}$. We construct a graph $G_k = (V, E)$ as follows. Start with $K_{3,3}$ and replace each edge by $t = 19k$ edge-disjoint paths of length 2. Note that the total number of edges is $|E| = 9 \cdot 2t = 18t$. Graph G_k is 3-quasiplanar. Since $\text{cr}(K_{3,3}) = 1$, it admits a drawing with precisely one crossing. The paths of length 2 can be drawn close to the edges of $K_{3,3}$ such that two paths cross if and only if the two corresponding edges of $K_{3,3}$ cross. Consequently any two crossing edges in this drawing are part of two paths that correspond to two crossing edges of $K_{3,3}$, which in turn implies that no three edges of G_k pairwise cross.

Suppose that G_k admits a k -gap-planar drawing Γ . Then the total number of crossings is at most $k|E| = 18kt$. We derive a contradiction by showing that $\text{cr}(G_k) \geq 19kt$. If we choose one of the t paths for each of the 9 edges of $K_{3,3}$ independently, then we obtain a subdivision of $K_{3,3}$, therefore there is a crossing between at least one pair of paths. There are t^9 ways to choose a path for each of the 9 edges of $K_{3,3}$. Each crossing between two paths in Γ is counted $t^{9-2} = t^7$ times. Consequently, the total number of crossings in Γ is at least $t^2 = 19kt$. \square

We now show that every $2k$ -planar drawing is k -gap-planar. We note that a similar result can be also derived from [16] (Lemma 10) for the case $k = 1$. A bipartite graph with vertex sets A and B is denoted as $H = (A, B, E)$. A *matching* from A into B is a set $M \subseteq E$ such that every vertex in A is incident to exactly one edge in M and every vertex in B is incident to at most one edge in M . The *neighborhood* of a subset $A' \subseteq A$ is the set of all vertices in B that are adjacent to a vertex in A' , and is denoted as $N(A')$. We recall that, by Hall's theorem, the graph H has a matching from A into B if and only if for each set $A' \subseteq A$ it is $|N(A')| \geq |A'|$.

Lemma 6. *For all $k \geq 1$, every $(2k)$ -planar drawing is k -gap-planar.*

Proof. Let G be a $(2k)$ -planar graph, for any $k \geq 1$, and let Γ be a $2k$ -planar drawing of G . Let $H = (A \cup B, E_H)$ be a bipartite graph obtained as follows. The set A has a vertex $a_{e,f}$ for each crossing in Γ between two edges e and f of G . For each edge e of G there are k vertices b_e^1, \dots, b_e^k in B . For every pair of edges e, f of G that cross in Γ , graph H contains edges $(a_{e,f}, b_e^1), \dots, (a_{e,f}, b_e^k)$ and $(a_{e,f}, b_f^1), \dots, (a_{e,f}, b_f^k)$ in H . Notice that if H admits a matching of A in B , then each crossing of Γ between an edge e and an edge f of G can be assigned to either e or f , and no edge of G is assigned with more than k crossings.

Consider any subset A' of A , and let $B' = N(A')$ be the neighborhood of A' in B . We claim that $|A'| \leq |B'|$. Let $E' \subseteq E_H$ denote the edges between A' and B' . By construction every vertex in A has degree $2k$, and hence $|E'| \geq 2k|A'|$. On the other hand, every vertex in B has degree at most $2k$ as every edge of G has at most $2k$ crossings, and hence $|E'| \leq 2k|B'|$. Hence $|A'| \leq |B'|$ as claimed.

By Hall's theorem, it now follows that H admits a matching from A into B , which corresponds to an assignment of gaps in Γ such that no edge has more than k gaps, i.e., Γ is a k -gap-planar drawing. \square

To conclude the proof of Theorem 7, we should prove that for every $k \geq 1$, there is a k -gap-planar graph that is not $2k$ -planar. A stronger result holds:

Lemma 7. *For every $k \geq 1$, there exists a 1-gap-planar graph G_k that is not k -planar.*

7 Conclusions and Open Problems

We introduced k -gap-planar graphs, our results give rise to several questions for future research. Among them are: (i) We proved that k -gap-planar graphs have $O(\sqrt{k} \cdot n)$ edges, and that 1-gap-planar graphs have at most $5n - 10$ edges, which is a tight bound. Can we establish a tight bound also for 2-gap-planar graphs? (ii) We proved that K_n is 1-gap-planar if and only if $n \leq 8$. A similar characterization could be studied also for complete bipartite graphs. Note that $K_{5,7}$ is not 1-gap-planar since it has crossing number greater than its number of edges, while we can exhibit a 1-gap-planar drawing of $K_{5,6}$. It is open whether $K_{6,6}$ is 1-gap-planar. Similarly, $K_{3,12}$ (Fig. 3(b)) and $K_{4,8}$ are 1-gap-planar, while we ask if this is true also for $K_{3,13}$ and $K_{4,9}$. (iii) We proved that deciding whether a graph is 1-gap-planar is NP-complete, even if the rotation system is fixed. Does the problem become polynomial for drawings in which all vertices are on the outer boundary? (iv) We proved that a drawing with at most $2k$ crossings per edge is k -gap-planar, and that a k -gap-planar drawing does not contain $2k + 2$ pairwise crossing edges. Do 1-gap-planar graphs have RAC drawings with at most 1 or 2 bends per edge?

References

1. Ackerman, E., Keszegh, B., Vizer, M.: On the size of planarly connected crossing graphs. In: Hu, Y., Nöllenburg, M. (eds.) GD 2016. LNCS, vol. 9801, pp. 311–320. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_24
2. Ackerman, E., Tardos, G.: On the maximum number of edges in quasi-planar graphs. *J. Combin. Theory Ser. A* **114**(3), 563–571 (2007)
3. Agarwal, P.K., Aronov, B., Pach, J., Pollack, R., Sharir, M.: Quasi-planar graphs have a linear number of edges. *Combinatorica* **17**(1), 1–9 (1997)
4. Angelini, P., et al.: On the relationship between k -planar and k -quasi-planar graphs. In: Bodlaender, H.L., Woeginger, G.J. (eds.) WG 2017. LNCS, vol. 10520, pp. 59–74. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68705-6_5
5. Appel, A., Rohlf, F.J., Stein, A.J.: The haloed line effect for hidden line elimination. *SIGGRAPH Comput. Graph.* **13**(2), 151–157 (1979)
6. Auer, C., Bachmaier, C., Brandenburg, F.J., Gleißner, A., Hanauer, K., Neuwirth, D., Reislhuber, J.: Outer 1-planar graphs. *Algorithmica* **74**(4), 1293–1320 (2016)
7. Auer, C., Brandenburg, F.J., Gleißner, A., Reislhuber, J.: 1-planarity of graphs with a rotation system. *J. Graph Algorithms Appl.* **19**(1), 67–86 (2015)
8. Bae, S.W., Baffier, J.F., Chun, J., Eades, P., Eickmeyer, K., Grilli, L., Hong, S.H., Korman, M., Montecchiani, F., Rutter, I., Tóth, C.D.: Gap-planar graphs. *CoRR abs/1708.07653* (2017). <https://arxiv.org/abs/1708.07653>
9. Bekos, M.A., Cornelsen, S., Grilli, L., Hong, S.H., Kaufmann, M.: On the recognition of fan-planar and maximal outer-fan-planar graphs. *Algorithmica* **79**, 1–27 (2016)

10. Bekos, M.A., Didimo, W., Liotta, G., Mehrabi, S., Montecchiani, F.: On RAC drawings of 1-planar graphs. *Theor. Comput. Sci.* **689**, 48–57 (2017)
11. Bekos, M.A., Kaufmann, M., Raftopoulos, C.N.: On the density of non-simple 3-planar graphs. In: Hu, Y., Nöllenburg, M. (eds.) GD 2016. LNCS, vol. 9801, pp. 344–356. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_27
12. Binucci, C., Di Giacomo, E., Didimo, W., Montecchiani, F., Patrignani, M., Symvonis, A., Tollis, I.G.: Fan-planarity: properties and complexity. *Theor. Comput. Sci.* **589**, 76–86 (2015)
13. Brandenburg, F.J., Didimo, W., Evans, W.S., Kindermann, P., Liotta, G., Montecchiani, F.: Recognizing and drawing IC-planar graphs. *Theor. Comput. Sci.* **636**, 1–16 (2016)
14. Brinkmann, G., Goedgebeur, J., McKay, B.D.: The generation of fullerenes. *J. Chem. Inf. Model.* **52**(11), 2910–2918 (2012)
15. Bruckdorfer, T., Cornelsen, S., Gutwenger, C., Kaufmann, M., Montecchiani, F., Nöllenburg, M., Wolff, A.: Progress on partial edge drawings. In: Didimo, W., Patrignani, M. (eds.) GD 2012. LNCS, vol. 7704, pp. 67–78. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36763-2_7
16. Bruckdorfer, T., Kaufmann, M.: Mad at edge crossings? Break the edges!. In: Kranakis, E., Krizanc, D., Luccio, F. (eds.) FUN 2012. LNCS, vol. 7288, pp. 40–50. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30347-0_7
17. Cheong, O., Har-Peled, S., Kim, H., Kim, H.-S.: On the number of edges of fan-crossing free graphs. In: Cai, L., Cheng, S.-W., Lam, T.-W. (eds.) ISAAC 2013. LNCS, vol. 8283, pp. 163–173. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45030-3_16
18. Dehkordi, H.R., Eades, P., Hong, S., Nguyen, Q.H.: Circular right-angle crossing drawings in linear time. *Theor. Comput. Sci.* **639**, 26–41 (2016)
19. Didimo, W., Liotta, G.: The crossing angle resolution in graph drawing. In: Pach, J. (ed.) *Thirty Essays on Geometric Graph Theory*. Springer, New York (2012). https://doi.org/10.1007/978-1-4614-0110-0_10
20. Eades, P., Hong, S., Katoh, N., Liotta, G., Schweitzer, P., Suzuki, Y.: A linear time algorithm for testing maximal 1-planarity of graphs with a rotation system. *Theor. Comput. Sci.* **513**, 65–76 (2013)
21. Eppstein, D., van Kreveld, M.J., Mumford, E., Speckmann, B.: Edges and switches, tunnels and bridges. *Comput. Geom.* **42**(8), 790–802 (2009)
22. Fox, J., Pach, J., Suk, A.: The number of edges in k -quasi-planar graphs. *SIAM J. Discr. Math.* **27**(1), 550–561 (2013)
23. Frank, A., Gyárfás, A.: How to orient the edges of a graph? *Colloq. Math. Soc. János Bolyai* **18**, 353–364 (1978)
24. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York (1979)
25. Grigoriev, A., Bodlaender, H.L.: Algorithms for graphs embeddable with few crossings per edge. *Algorithmica* **49**(1), 1–11 (2007)
26. Hoffmann, M., Tóth, C.D.: Two-planar graphs are quasiplanar. *CoRR* [abs/1705.05569](https://arxiv.org/abs/1705.05569) (2017). <http://arxiv.org/abs/1705.05569>
27. Hong, S., Eades, P., Katoh, N., Liotta, G., Schweitzer, P., Suzuki, Y.: A linear-time algorithm for testing outer-1-planarity. *Algorithmica* **72**(4), 1033–1054 (2015)
28. Hong, S.-H., Nagamochi, H.: Testing full outer-2-planarity in linear time. In: Mayr, E.W. (ed.) WG 2015. LNCS, vol. 9224, pp. 406–421. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53174-7_29

29. Hong, S.H., Kaufmann, M., Kobourov, S.G., Pach, J.: Beyond-planar graphs: algorithmics and combinatorics (Dagstuhl Seminar 16452). Dagstuhl Rep. **6**(11), 35–62 (2017)
30. Hong, S.H., Tokuyama, T.: Algorithmics for beyond planar graphs. In: NII Shonan Meeting, Shonan Village Center (2016). <http://shonan.nii.ac.jp/shonan/blog/2015/11/15/3972/>
31. Huang, W., Eades, P., Hong, S.: Larger crossing angles make graphs easier to read. *J. Vis. Lang. Comput.* **25**(4), 452–465 (2014)
32. Kaufmann, M., Ueckerdt, T.: The density of fan-planar graphs. CoRR abs/1403.6184 (2014). <http://arxiv.org/abs/org/abs/1403.6184>
33. Kleitman, D.J.: The crossing number of $K_{5,n}$. *J. Combin. Theor.* **9**(4), 315–323 (1970)
34. Kobourov, S.G., Liotta, G., Montecchiani, F.: An annotated bibliography on 1-planarity. *Comput. Sci. Rev.* **25**, 59–74 (2017). https://doi.org/10.1007/978-3-319-68705-6_5
35. Korzhik, V.P., Mohar, B.: Minimal obstructions for 1-immersions and hardness of 1-planarity testing. *J. Graph Theor.* **72**(1), 30–71 (2013)
36. Liotta, G.: Graph drawing beyond planarity: some results and open problems. In: ICTCS 2014. CEUR Workshop Proceedings, vol. 1231, pp. 3–8 (2014). [CEUR-WS.org](http://www.ceur-ws.org)
37. Pach, J., Radoičić, R., Tardos, G., Tóth, G.: Improving the crossing lemma by finding more crossings in sparse graphs. *Discrete Comput. Geom.* **36**(4), 527–552 (2006)
38. Pach, J., Tóth, G.: Graphs drawn with few crossings per edge. *Combinatorica* **17**(3), 427–439 (1997)
39. Picard, J.C., Queyranne, M.: A network flow solution to some nonlinear 0 – 1 programming problems, with applications to graph theory. *Networks* **12**(2), 141–159 (1982)
40. Venkateswaran, V.: Minimizing maximum indegree. *Discr. Appl. Math.* **143**(13), 374–378 (2004)
41. Zarankiewicz, C.: On a problem of P. Turan concerning graphs. *Fund. Math.* **41**(1), 137–145 (1955)

Beyond Outerplanarity

Steven Chaplick¹, Myroslav Kryven¹, Giuseppe Liotta², Andre Löffler¹ (✉),
and Alexander Wolff¹ (ID)

¹ Lehrstuhl für Informatik I, Universität Würzburg, Würzburg, Germany
andre.loffler@uni-wuerzburg.de

² Department of Engineering, University of Perugia, Perugia, Italy
giuseppe.liotta@unipg.it
<http://www1.informatik.uni-wuerzburg.de/en/staff>

Abstract. We study straight-line drawings of graphs where the vertices are placed in convex position in the plane, i.e., *convex drawings*. We consider two families of graph classes with nice convex drawings: *outer k -planar* graphs, where each edge is crossed by at most k other edges; and, *outer k -quasi-planar* graphs where no k edges can mutually cross.

We show that the outer k -planar graphs are $(\lfloor \sqrt{4k+1} \rfloor + 1)$ -degenerate, and consequently that every outer k -planar graph can be $(\lfloor \sqrt{4k+1} \rfloor + 2)$ -colored, and this bound is tight. We further show that every outer k -planar graph has a balanced separator of size at most $2k+3$. For each fixed k , these small balanced separators allow us to test outer k -planarity in quasi-polynomial time, i.e., none of these recognition problems are NP-hard unless ETH fails.

For the outer k -quasi-planar graphs we discuss the edge-maximal graphs which have been considered previously under different names. We also construct planar 3-trees that are not outer 3-quasi-planar.

Finally, we restrict outer k -planar and outer k -quasi-planar drawings to *closed* drawings, where the vertex sequence on the boundary is a cycle in the graph. For each k , we express *closed outer k -planarity* and *closed outer k -quasi-planarity* in *extended monadic second-order logic*. Thus, since outer k -planar graphs have bounded treewidth, closed outer k -planarity is linear-time testable by Courcelle's Theorem.

1 Introduction

A *drawing* of a graph maps each vertex to a distinct point in the plane, each edge to a Jordan curve connecting the points of its incident vertices but not containing the point of any other vertex, and two such Jordan curves have at most one common point. In the last few years, the focus in graph drawing has shifted from exploiting structural properties of planar graphs to addressing the question of how to produce well-structured (understandable) drawings in the presence of edge crossings, i.e., to the topic of *beyond-planar* graph classes. The primary approach here has been to define and study graph classes which allow

The full version of this paper is available at <http://arxiv.org/abs/1708.08723v2>.

some edge crossings, but restrict the crossings in various ways. Two commonly studied such graph classes are:

1. *k-planar graphs*, the graphs which can be drawn so that each edge (Jordan curve) is crossed by at most k other edges.
2. *k-quasi-planar graphs*, the graphs which can be drawn so that no k pairwise non-incident edges mutually cross.

Note that the 0-planar graphs and 2-quasi-planar graphs are precisely the planar graphs. Additionally, the 3-quasi-planar graphs are simply called *quasi-planar*.

In this paper we study these two families of classes of graphs by restricting the drawings so that the points are placed in convex position and edges mapped to line segments, i.e., we apply the above two generalizations of planar graphs to outerplanar graphs and study *outer k-planarity* and *outer k-quasi-planarity*. We consider balanced separators, treewidth, degeneracy (see paragraph “Concepts” below), coloring, edge density, and recognition for these classes.

Related work. Ringel [27] was the first to consider k -planar graphs by showing that 1-planar graphs are 7-colorable. This was later improved to 6-colorable by Borodin [8]. This is tight since K_6 is 1-planar. Many additional results on 1-planarity can be found in a recent survey paper [21]. Generally, each n -vertex k -planar graph has at most $4.108n\sqrt{k}$ edges [26] and treewidth $O(\sqrt{kn})$ [14].

Outer k -planar graphs have been considered mostly for $k \in \{0, 1, 2\}$. Of course, the outer 0-planar graphs are the classic outerplanar graphs which are well-known to be 2-degenerate and have treewidth at most 2. It was shown that essentially every graph property is testable on outerplanar graphs [5]. Outer 1-planar graphs are a simple subclass of planar graphs and can be recognized in linear time [4, 18]. *Full outer 2-planar graphs*, which form a subclass of outer 2-planar graphs, can be recognized in linear time [19]. General outer k -planar graphs were considered by Binucci et al. [7], who (among other results) showed that, for every k , there is a 2-tree which is not outer k -planar. Wood and Telle [30] considered a slight generalization of outer k -planar graphs in their work and showed that these graphs have treewidth $O(k)$.

The k -quasi-planar graphs have been heavily studied from the perspective of edge density. The goal here is to settle a conjecture of Pach et al. [25] stating that every n -vertex k -quasi-planar graph has at most $c_k n$ edges, where c_k is a constant depending only on k . This conjecture is true for $k = 3$ [2] and $k = 4$ [1]. The best known upper bound is $(n \log n)2^{\alpha(n)^{c_k}}$ [16], where α is the inverse of the Ackermann function. Edge density was also considered in the “outer” setting: Capocelas and Pach [9] showed that any k -quasi-planar graph has at most $2(k-1)n - \binom{2k-1}{2}$ edges, and that there are k -quasi-planar graphs meeting this bound. More recently, it was shown that the *semi-bar k-visibility graphs* are outer $(k+2)$ -quasi-planar [17]. However, the outer k -quasi-planar graph classes do not seem to have received much further attention.

The relationship between k -planar graphs and k -quasi-planar graphs was considered recently. While any k -planar graph is clearly $(k+2)$ -quasi-planar, Angelini et al. [3] showed that any k -planar graph is even $(k+1)$ -quasi-planar.

The *convex* (or *1-page book*) *crossing number* of a graph [29] is the minimum number of crossings which occur in any convex drawing. This concept has been introduced several times (see [29] for more details). The convex crossing number is NP-complete to compute [23]. However, recently Bannister and Eppstein [6] used treewidth-based techniques (via extended monadic second order logic) to show that it can be computed in linear FPT time, i.e., $O(f(c) \cdot n)$ time where c is the convex crossing number and f is a computable function. Thus, for any k , the *outer k -crossing graphs* can be recognized in time linear in $n + m$.

Concepts. We briefly define the key graph theoretic concepts that we will study.

A graph is *d -degenerate* when every subgraph of it has a vertex of degree at most d . This concept was introduced as a way to provide easy coloring bounds [22]. Namely, a d -degenerate graph can be inductively $d + 1$ colored by simply removing a vertex of degree at most d . A graph class is *d -degenerate* when every graph in the class is d -degenerate. Furthermore, a graph class which is *hereditary* (i.e., closed under taking subgraphs) is *d -degenerate* when every graph in that class has a vertex of degree at most d . Note that outerplanar graphs are 2-degenerate, and planar graphs are 5-degenerate.

A *separation* of a graph G is pair A, B of subsets of $V(G)$ such that $A \cup B = V(G)$, and no edge of G has one end in $A \setminus B$ and the other in $B \setminus A$. The set $A \cap B$ is called a *separator* and the *size* of the separation (A, B) is $|A \cap B|$. A separation (A, B) of a graph G on n vertices is *balanced* if $|A \setminus B| \leq \frac{2n}{3}$ and $|B \setminus A| \leq \frac{2n}{3}$. The *separation number* of a graph G is the smallest number s such that every subgraph of G has a balanced separation of size at most s . The *treewidth* of a graph was introduced by Robertson and Seymour [28]; it is closely related to the separation number. Namely, any graph with treewidth t has separation number at most $t + 1$ and, as Dvořák and Norin [15] recently showed, any graph with separation number s has treewidth at most $105s$. Graphs with bounded treewidth are well-known due to Courcelle's Theorem (see Theorem 6) [10], i.e., having bounded treewidth means many problems can be solved efficiently.

The *Exponential Time Hypothesis (ETH)* [20] is a complexity theoretic assumption defined as follows. For $k \geq 3$, let $s_k = \inf\{\delta : \text{there is an } O(2^{\delta n})\text{-time algorithm to solve } k\text{-SAT}\}$. ETH states that for $k \geq 3$, $s_k > 0$, e.g., there is no *quasi-polynomial time*¹ algorithm that solves 3-SAT. So, finding a problem that can be solved in quasi-polynomial time and is also NP-complete, would contradict ETH. In recent years, ETH has become a standard assumption from which many conditional lower bounds have been proven [12].

Contribution. In Sect. 2, we consider outer k -planar graphs. We show that they are $(\lfloor \sqrt{4k + 1} \rfloor + 1)$ -degenerate, and observe that the largest outer k -planar clique has size $(\lfloor \sqrt{4k + 1} \rfloor + 2)$, i.e., implying each outer k -planar graph can be $(\lfloor \sqrt{4k + 1} \rfloor + 2)$ -colored and this is tight. We further show that every outer k -planar graph has separation number at most $2k + 3$. For each fixed k , we use these balanced separators to obtain a quasi-polynomial time algorithm to test outer k -planarity, i.e., these recognition problems are not NP-hard unless ETH fails.

¹ i.e., with a runtime of the form $2^{\text{poly}(\log n)}$.

In Sect. 3, we consider outer k -quasi-planar graphs. Specifically, we discuss the edge-maximal graphs which have been considered previously under different names [9, 13, 24]. We also relate outer k -quasi-planar graphs to planar graphs.

Finally, in Sect. 4, we restrict outer k -planar and outer k -quasi-planar drawings to *closed* drawings, where the sequence of vertices on the outer boundary is a cycle. For each k , we express both *closed outer k -planarity* and *closed outer k -quasi-planarity* in *extended monadic second-order logic*. Thus, closed outer k -planarity is testable in $O(f(k) \cdot n)$ time, for a computable function f .

2 Outer k -Planar Graphs

In this section we show that every outer k -planar graph is $O(\sqrt{k})$ -degenerate and has separation number $O(k)$. This provides tight bounds on the chromatic number, and allows for testing outer k -planarity in quasi-polynomial time.

Degeneracy. We show that every outer k -planar graph has a vertex of degree at most $\sqrt{4k + 1} + 1$. First we note the size of the largest outer k -planar clique and then we prove that each outer k -planar graph has a vertex matching the clique’s degree. This also tightly bounds the chromatic number in terms of k , i.e., Theorem 1 follows from Lemma 1 (proven in Appendix B.1) and Lemma 2.

Lemma 1. *Every outer k -planar clique has at most $\lfloor \sqrt{4k + 1} \rfloor + 2$ vertices.*

Lemma 2. *An outer k -planar graph can have maximum minimum degree at most $\sqrt{4k + 1} + 1$ and this bound is tight.*

Proof. Assume that the outer k -planar graph has maximum minimum degree δ . Since we can create a clique with $\lfloor \sqrt{4k + 1} \rfloor + 2$ vertices (see Lemma 1), $\delta \geq \lfloor \sqrt{4k + 1} \rfloor + 1$. Let us show that δ cannot be larger than $\sqrt{4k + 1} + 1$.

Consider an edge ab that cuts $l \in \mathbb{N}$ vertices of the graph to one side (not counting a and b), then there are at least $\delta l - l(l + 1)$ edges crossing the edge ab . We will now show by induction that if there existed an outer k -planar graph with minimum degree $\delta \geq \sqrt{4k + 1} + 2$, it would be too small to accommodate such a minimum degree vertex.

Any edge ab that cuts l vertices is crossed by at least $\delta l - l(l + 1)$ edges. Therefore, if $\delta \geq \sqrt{4k + 1} + 2$, there is l^* such that ab cannot cut $l^* \geq \frac{1}{2}(\delta - 1 - \sqrt{(\delta - 1)^2 - 4(k + 1)})$ vertices because then it is crossed by $\delta l^* - l^*(l^* + 1) \geq k + 1$ edges. Take the smallest such l^* and let us show that there also cannot be an edge ab that cuts more than l^* vertices. As the induction hypothesis, assume that no edge ab cuts between l^* and l vertices inclusive. Thus, the minimum number of edges that cross ab is: $\delta l - l(l + 1) + 2(\sum_{j=1}^{l-l^*} j) > k$, where the last term accounts for the absent edges that cut more than $l - l^*$ vertices. Now, if ab cuts $l + 1$ vertices, it is crossed by

$$\begin{aligned} &\geq \delta l - l(l + 1) + 2(\sum_{j=1}^{l-l^*} j) + \delta - 2(l + 1) + 2(l - l^* + 1) \\ &> k + \delta - 2(l + 1) + 2(l - l^* + 1) > k \end{aligned}$$

edges if $\delta > 2l^*$.

Since for $\delta > \sqrt{4k+1} + 2$ the inequality is always satisfied, there cannot be an edge that cuts more than $l^* < \sqrt{4k+1}/2$ vertices in any outer k -planar graph with the maximum minimum degree $\delta \geq \sqrt{4k+1} + 2$. But then, such a graph can have at most $2l^* < \sqrt{4k+1}$ vertices, which is not enough to accommodate the minimum degree vertex required; a contradiction. \square

Theorem 1. *Each outer k -planar graph is $\sqrt{4k+1} + 2$ colorable. This is tight.*

Quasi-polynomial time recognition via balanced separators. We show that outer k -planar graphs have separation number at most $2k+3$ (Theorem 2). Via a result of Dvořák and Norin [15], this implies they have $O(k)$ treewidth. However, Proposition 8.5 of [30] implies that every outer k -planar graph has treewidth at most $3k + 11$, i.e., a better bound on the treewidth than applying the result of Dvořák and Norin to our separators. The treewidth $3k + 11$ bound also implies a separation number of $3k + 12$, but our bound is better. Our separators also allow outer k -planarity testing in quasi-polynomial time (Theorem 3).

Theorem 2. *Each outer k -planar graph has separation number at most $2k + 3$.*

Proof. Consider an outer k -planar drawing. If the graph has an edge that cuts $[\frac{n}{3}, \frac{2n}{3}]$ vertices to one side, we can use this edge to obtain a balanced separator of size at most $k + 2$, i.e., by choosing the endpoints of this edge and a vertex cover of the edges crossing it. So, suppose no such edge exists. Consider a pair of vertices (a, b) such that the line ab divides the drawing into left and right sides having an almost equal number of vertices (with a difference at most one). If the edges which cross the line ab also mutually cross each other, there can be at most k of them. Thus, we again have a balanced separator of size at most $k + 2$. So, it remains to consider the case when we have a pair of edges that cross the line ab , but do not cross each other. We call such a pair of edges *parallel*. We now pick a pair of parallel edges in a specific way. Starting from b , let b_l be the first vertex along the boundary in clockwise direction such that there is an edge $b_l b'_l$ that crosses the line ab . Symmetrically, starting from a , let a_r be the first vertex along the boundary in clockwise direction such that there is an edge $a_r a'_r$ that crosses the line ab ; see Fig. 1 (left). Note that the edges $a_r a'_r$ and $b_l b'_l$ are either identical or parallel. In the former case, we see that all other edges crossing the line ab must also cross the edge $a_r a'_r = b_l b'_l$, and as such there are again at most k edges crossing the line ab . In the latter case, there are two subcases that we treat below. For two vertices u and v , let $[u, v]$ be the set of vertices that starts with u and, going clockwise, ends with v . Let $(u, v) = [u, v] \setminus \{u, v\}$.

Case 1. *The edge $b_l b'_l$ cuts $\mu \leq \frac{n}{3}$ vertices to the top; see Fig. 1 (center).*

In this case, either $[b'_l, b]$ or $[b, b_l]$ has $[\frac{n}{3}, \frac{n}{2}]$ vertices. We claim that neither the line bb_l nor the line bb'_l can be crossed more than k times. Namely, each edge that crosses the line bb_l also crosses the edge $b_l b'_l$. Similarly, each edge that crosses the line bb'_l also crosses the edge $b_l b'_l$. Thus, we have a separator of size at most $k + 2$, regardless of whether we choose bb_l or bb'_l to separate the graph. As we observed above, one of them is balanced.

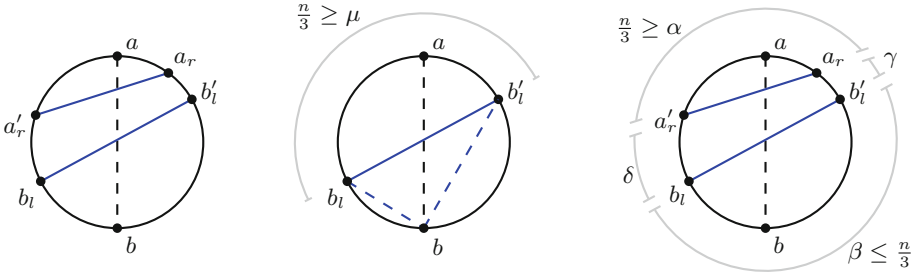


Fig. 1. Left: the pair of parallel edges $b_l b'_l$ and $a_r a'_r$; center: case 1; right: case 2

Case 1'. The edge $a_r a'_r$ cuts at most $\frac{n}{3}$ vertices to the bottom.

This is symmetric to case 1.

Case 2. The edge $b_l b'_l$ cuts at most $\frac{n}{3}$ vertices to the bottom, and the edge $a_r a'_r$ cuts at most $\frac{n}{3}$ vertices to the top; see Fig. 1 (right).

We show that we can always find a pair of parallel edges such that one cuts at most $\frac{n}{3}$ vertices to the bottom and the other cuts at most $\frac{n}{3}$ vertices to the top, and no edge between them is parallel to either of them. We call such a pair *close*. If there is an edge e between $b_l b'_l$ and $a_r a'_r$, we form a new pair by using e and $a_r a'_r$ if e cuts at most $\frac{n}{3}$ vertices to the bottom or by using e and $b_l b'_l$ if e cuts at most $\frac{n}{3}$ vertices to the top. By repeating this procedure, we always find a close pair. Hence, we can assume that $b_l b'_l$ and $a_r a'_r$ actually form a close pair. Let $\alpha = |(a'_r, a_r)|$, $\beta = |(b'_l, b_l)|$, $\gamma = |(a_r, b'_l)|$, and $\delta = |(b_l, a'_r)|$; see Fig. 1 (right).

Suppose that $a'_r = b_l$ or $a_r = b'_l$. We can now use both edges $b_l b'_l$ and $a_r a'_r$ (together with any edges crossing them) to obtain a separator of size at most $2k + 3$. The separator is balanced since $\alpha + \beta \leq \frac{2n}{3}$ and $\gamma + \delta \leq \frac{2n}{3}$.

So, now a_r, a'_r, b_l, b'_l are all distinct. Note that $\gamma, \delta \leq \frac{n}{2}$ since each side of the line ab has at most $\frac{n}{2}$ vertices. We separate the graph along the line $b_l a_r$. Namely, all the edges that cross this line must also cross $b_l b'_l$ or $a'_r a_r$. Therefore, we obtain a separator of size at most $2k + 2$.

To see that the separator is balanced, we consider two cases. If $\delta \geq \frac{n}{3}$ (or $\gamma \geq \frac{n}{3}$), then $\alpha + \beta + \gamma \leq \frac{2n}{3}$ (or $\alpha + \beta + \delta \leq \frac{2n}{3}$). Otherwise $\delta < \frac{n}{3}$ and $\gamma < \frac{n}{3}$. In this case $\delta + \alpha \leq \frac{2n}{3}$ and $\gamma + \beta \leq \frac{2n}{3}$. In both cases the separator is balanced. \square

Theorem 3. For fixed k , testing the outer k -planarity of an n -vertex graph takes $O(2^{\text{polylog } n})$ time.

Proof. Our approach is to leverage the structure of the balanced separators as described in the proof of Theorem 2. Namely, we enumerate the sets which could correspond to such a separator, pick an appropriate outer k -planar drawing of these vertices and their edges, partition the components arising from this separator into *regions*, and recursively test the outer k -planarity of the regions.

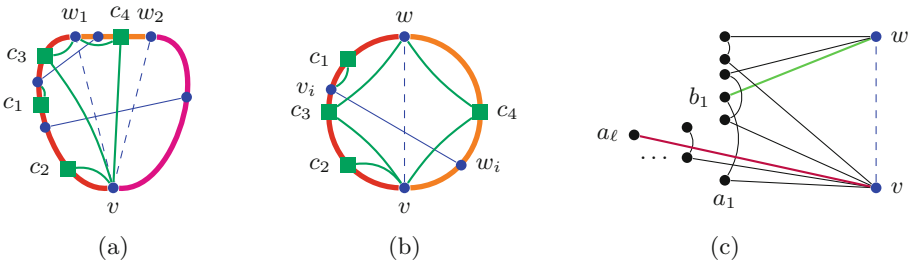


Fig. 2. Shapes of separators, special separator S in blue, regions in different colors (red, orange, and pink), components connected to blue vertices in green: (a) closest-parallel case; (b) single-edge case; (c) special case for single-edge separators. (Color figure online)

To obtain quasi-polynomial runtime, we need to limit the number of components on which we branch. To do so, we group them into regions defined by special edges of the separators.

By the proof of Theorem 2, if our input graph has an outer k -planar drawing, there must be a separator which has one of the two shapes depicted in Fig. 2(a) and (b). Here we are not only interested in the up to $2k + 3$ vertices of the balanced separator, but actually the set S of up to $4k + 3$ vertices one obtains by taking both endpoints of the edges used to find the separator. Note: S is also a balanced separator. We use a brute force approach to find such an S . Namely, we first enumerate vertex sets of size up to $4k + 3$. We then consider two possibilities, i.e., whether this set can be drawn similar to one of the two shapes from Fig. 2. So, we now fix this set S . Note that since S has $O(k)$ vertices, the subgraph G_S induced by S can have at most a function of k different outer k -planar drawings. Thus, we further fix a particular drawing of G_S .

We now consider the two different shapes separately. In the first case, in S , we have three special vertices v, w_1 and w_2 and in the second case we will have two special vertices v and w . These vertices will be called *boundary* vertices and all other vertices in S will be called *regional* vertices. Note that, since we have a fixed drawing of G_S , the regional vertices are partitioned into regions by the specially chosen boundary vertices. Now, from the structure of the separator which is guaranteed by the proof of Theorem 2, no component of $G \setminus S$ can be adjacent to regional vertices which live in different regions with respect to the boundary vertices.

We first discuss the case of using G_S as depicted in Fig. 2(a). Here, we start by picking the three special vertices v, w_1 and w_2 from S to take the role as shown in Fig. 2(a). The following arguments regarding this shape of separator are symmetric with respect to the pair of opposing regions.

Notice that if there is a component connected to regional vertices of different regions, we can reject this configuration. From the proof of Theorem 2, we further observe that no component can be adjacent to all three boundary vertices. Namely, this would contradict the closeness of the parallel edges or it would

contradict the members of the separator, i.e., it would imply an edge connecting distinct regions. We now consider the four possible different types of components c_1, c_2, c_3 and c_4 in Fig. 2(a) that can occur in a region neighboring w_1 . Components of type c_1 are connected to (possibly many) regional vertices of the same region and may be connected to boundary vertices as well. In any valid drawing, they will end up in the same region as their regional vertices. Components of type c_2 are not connected to any regional vertices and only connected to one of the three boundary vertices. Since they are not connected to regional vertices, they can not interfere with other parts of the drawing, so we can arbitrarily assign them to an adjacent region of their boundary vertex. Components that are connected to two boundary vertices appear at first to have two possible placements, e.g., as c_3 or c_4 in Fig. 2(a). However, c_4 is not a valid placement for this type of component since it would contradict the fact that this separator arose from two close parallel edges as argued in the proof of Theorem 2. From the above discussion, we see that from a fixed configuration (i.e., set S , drawing of G_S , and triple of boundary vertices), if the drawing of $G \setminus S$ has the shape depicted in Fig. 2(a), we can either reject the current configuration (based on having bad components), or we see that every component of $G \setminus S$ is either attached to exactly one boundary vertex or it has a well-defined placement into the regions defined by the boundary vertices. For those components which are attached to exactly one boundary vertex, we observe that it suffices to recursively produce a drawing of that component together with its boundary vertex and to place this drawing next to the boundary vertex. For the other components, we partition them into their regions and recurse on the regions. This covers all cases for this separator shape.

The other shape of our separator can be seen in Fig. 2(b). Note that we now have two boundary vertices v and w and thus only have two regions. Again we see the two component types c_1 and c_2 and can handle them as above. We also have components connected to both v and w but no regional vertices. These components now truly have two different placement options c_3, c_4 . If we have an edge $v_i w_i$ (as in Fig. 2(b)) of the separator that is not vw , we now observe that there cannot be more than k such components. Namely, in any drawing, for each component, there will be an edge connecting this component to either v or w which crosses $v_i w_i$. Thus, we now enumerate all the different placements of these components as type c_3 or c_4 and recurse accordingly.

However, the separator may be exactly the pair (v, w) . Note that there are no components of type c_1 and the components of type c_2 can be handled as before. We will now argue that we can have at most a function of k different components of type c_3 or c_4 in a valid drawing. Consider the components of type c_3 (the components of type c_4 can be counted similarly). In a valid drawing, each type c_3 component defines a sub-interval of the left region spanning from its highest to its lowest vertex such that these vertices are adjacent to one of v or w . Two such intervals relate in one of three ways: They overlap, they are disjoint, or one is contained in the other. We group components with either overlapping or disjoint intervals into *layers*. We depict this situation in Fig. 2(c) where, for simplicity,

for every component we only draw its highest vertex and its lowest vertex and they are connected by one edge.

Let a_1b_1 be the bottommost component of type c_3 (i.e., a_1 is the clockwise-first vertex from v in a component of type c_3). The first layer is defined as the component a_1b_1 together with every component whose interval either overlaps or is disjoint from the interval of a_1b_1 . Now consider the green edge b_1w (see Fig. 2(c)), note we may have that this edge connects a_1 to w instead. Now, for every component of this layer which is disjoint from the interval of a_1b_1 , this edge is crossed by at least one edge connecting it to v . Furthermore, for every component of this layer which overlaps the interval of a_1b_1 , there is an edge connecting b_1 to either v or w which is crossed by at least one edge within that component. So in total, there can only be $O(k)$ components in this first layer. New layers are defined by considering components whose intervals are contained in a_1b_1 . To limit the total number of layers, let a_ℓ be the bottommost vertex of the first component of the deepest layer and consider the purple edge va_ℓ . This edge is crossed by some edge of every layer above it and as any edge can only have k crossings, there can only be $O(k)$ different levels in total. This leaves us with a total of at most $O(k^2)$ components per region and again we can enumerate their placements and recurse accordingly.

The above algorithm provides the following recurrence regarding its runtime. Namely, we let $T(n)$ denote the runtime of our algorithm, and we can see that the following expression generously upper bounds its value. Here $f(s)$ denotes the number of different outer k -planar drawings of a graph with s vertices.

$$T(n) \leq \begin{cases} n^{O(k)} \cdot f(4k + 3) \cdot n^3 \cdot n \cdot T(\frac{2n}{3}) & \text{for } n > 5k \\ f(n) & \text{otherwise} \end{cases}$$

Thus, the algorithm runs in quasi-polynomial time, i.e., $2^{\text{poly}(\log n)}$. □

3 Outer k -Quasi-Planar Graphs

In this section we consider outer k -quasi-planar graphs. We first describe some classes of graphs which are outer 3-quasi-planar. We then discuss edge-maximal outer k -quasi-planar drawings.

Note, all sub-Hamiltonian planar graphs are outer 3-quasi-planar. One can also see which complete and bipartite complete graphs are outer 3-quasi-planar.

Proposition 1. *The following graphs are outer 3-quasi-planar: (a) $K_{4,4}$; (b) K_5 ; (c) planar 3-tree with three complete levels; (d) square-grids of any size.*

Proof. (a) and (b) are easily observed. (c) was experimentally verified by constructing a Boolean expression and using MiniSat to check it for satisfiability; see Appendix A. (d) follows from square-grids being sub-Hamiltonian. □

Correspondingly, we note complete and complete bipartite graphs which are not outer-quasi planar. Furthermore, not all planar graphs are outer quasi-planar, e.g., the vertex-minimal planar 3-tree in Fig. 3(a) is not outer quasi-planar, this was verified checking for satisfiability the corresponding Boolean

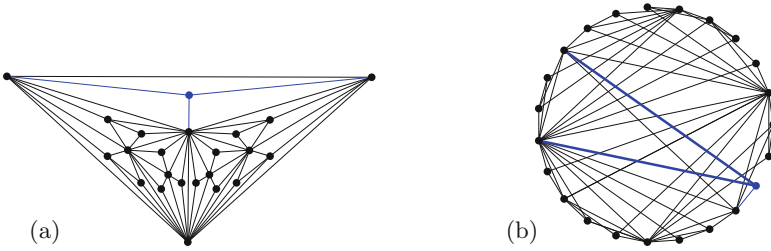


Fig. 3. A vertex-minimal 23-vertex planar 3-tree which is not outer quasi-planar:(a) planar drawing; (b) deleting the blue vertex makes the drawing outer quasi-planar (Color figure online)

expression; see Appendix A. A drawing of the graph in Fig. 3(b) was constructed by removing the blue vertex and drawing the remaining graph in an outer quasi-planar way.

Proposition 2. *The following graphs are not outer 3-quasi-planar: (a) $K_{p,q}$, $p \geq 3, q \geq 5$; (b) K_n , $n \geq 6$; (c) planar 3-tree with four complete levels.*

Together, Propositions 1 and 2 immediately yield the following.

Theorem 4. *Planar graphs and outer 3-quasi-planar graphs are incomparable under containment.*

Remark 1. For outer k -quasi-planar graphs ($k > 3$) containment questions become more intricate. Every planar graph is outer 5-quasi-planar because planar graphs have page number 4 [31]. We also know a planar graph that is not outer 3-quasi-planar. It is open whether every planar graph is outer 4-quasi-planar.

Maximal outer k -quasi-planar graphs. A drawing of an outer k -quasi-planar graph is called *maximal* if adding any edge to it destroys the outer k -quasi-planarity. We call an outer k -quasi-planar graph maximal if it has a maximal outer k -quasi-planar drawing. Recall that Capoleas and Pach [9] showed the following upper bound on the edge density of outer k -quasi-planar graphs on n vertices: $|E| \leq 2(k - 1)n - \binom{2k-1}{2}$.

We prove (see Appendix B.2) that each maximal outer k -quasi-planar graph meets this bound. Our proof builds on the ideas of Capoleas and Pach [9] and directly shows the result via an inductive argument. However, while preparing the camera-ready version of this paper, we learned of two other proofs of this result in the literature [13,24]. We thank David Wood for pointing us to these results. Both papers prove a slightly stronger theorem (concerning *edge flips*) as their main result. Namely, for a drawing $G = (V, E)$, an *edge flip* produces a new drawing G^* by replacing an edge $e \in E$ with a new edge $e^* \in \binom{V}{2} \setminus E$. They [13,24] show that, for every two maximal outer k -quasi-planar drawings $G = (V, E)$ and $G' = (V, E')$, there is a sequence of edge flips producing drawings $G = G_1, G_2, \dots, G_t = G'$ such that each G_i is a maximal k -quasi-planar drawing.

Together with the tight example of Capoyleas and Pach [9], this implies the next theorem, and makes our proof fairly redundant.

Theorem 5 ([13, 24]). *Each maximal outer k -quasi-planar drawing $G = (V, E)$ has:*

$$|E| = \begin{cases} \binom{|V|}{2} & \text{if } |V| \leq 2k - 1, \\ 2(k - 1)|V| - \binom{2k-1}{2} & \text{if } |V| \geq 2k - 1. \end{cases}$$

4 Closed Convex Drawings in MSO_2

Here we express graph properties in *extended monadic second-order logic* (MSO_2). This subset of *second-order logic* is built from the following primitives.

- variables for vertices, edges, sets of vertices, and sets of edges;
- binary relations for: equality ($=$), membership in a set (\in), subset of a set (\subseteq), and edge–vertex incidence (I);
- standard propositional logic operators: $\neg, \wedge, \vee, \rightarrow$.
- standard quantifiers (\forall, \exists) which can be applied to all types of variables.

For a graph G and an MSO_2 formula ϕ , we use $G \models \phi$ to indicate that ϕ can be satisfied by G in the obvious way. Properties expressed in this logic allow us to use the powerful algorithmic result of Courcelle stated next.

Theorem 6 ([10, 11]). *For any integer $t \geq 0$ and any MSO_2 formula ϕ of length ℓ , an algorithm can be constructed which takes a graph G with treewidth at most t and decides in $O(f(t, \ell) \cdot (n + m))$ time whether $G \models \phi$ where the function f from this time bound is a computable function of t and ℓ .*

Outer k -planar graphs are known to have treewidth $O(k)$ (see Proposition 8.5 of [30]). So, expressing outer k -planarity by an MSO_2 formula whose size is a function of k would mean that outer k -planarity could be tested in linear time. However, this task might be out of the scope of MSO_2 . The challenge in expressing outer k -planarity in MSO_2 is that MSO_2 does not allow quantification over sets of pairs of vertices which involve non-edges. Namely, it is unclear how to express a set of pairs that forms the circular order of vertices on the boundary of our convex drawing. However, if this circular order forms a *Hamiltonian cycle* in our graph, then we can indeed express this in MSO_2 . With the edge set of a Hamiltonian cycle of our graph in hand, we can then ask that this cycle was chosen in such a way that the other edges satisfy either k -planarity or k -quasi-planarity. With this motivation in mind, we define the classes *closed outer k -planar* and *closed outer k -quasi-planar*, where closed means that there is an appropriate convex drawing where the circular order forms a Hamiltonian cycle. Our main result here is stated next.

Theorem 7. *Closed outer k -planarity and closed outer k -quasi-planarity can be expressed in MSO_2 . Thus, closed outer k -planarity can be tested in linear time.*

The formulas for our graph properties are built using formulas for Hamiltonicity (HAMILTONIAN), partitioning of vertices into disjoint subsets (VERTEX-PARTITION) and connected induced subgraphs on sets of vertices using only a subset of the edges (CONNECTED). They can be found in Appendix C.

For a closed outer k -planar or closed outer k -quasi-planar graph G , we want to express that two edges e and e_i cross. To this end, we assume that there is a Hamiltonian cycle E^* of G that defines the outer face. We partition the vertices of G into three subsets C , A , and B , as follows: C is the set containing the endpoints of e , whereas A and B are connected subgraphs on the remaining vertices that use only edges of E^* . In this way, we partition the vertices of G into two sets, one left and the other one right of e . For such a partition, e_i must cross e whenever e_i has one endpoint in A and one in B .

$$\begin{aligned} \text{CROSSING}(E^*, e, e_i) &\equiv (\forall A, B, C) [(\text{VERTEX-PARTITION}(A, B, C) \\ &\wedge (I(e, x) \leftrightarrow x \in C) \wedge \text{CONNECTED}(A, E^*) \wedge \text{CONNECTED}(B, E^*)) \\ &\rightarrow (\exists a \in A)(\exists b \in B)[I(e_i, a) \wedge I(e_i, b)]] \end{aligned}$$

Now we can describe the crossing patterns for closed outer k -planarity and closed outer k -quasi-planarity as follows:

$$\begin{aligned} \text{CLOSED OUTER } k\text{-PLANAR}_G &\equiv (\exists E^*) \left[\text{HAMILTONIAN}(E^*) \wedge \right. \\ &(\forall e) \left[(\forall e_1, \dots, e_{k+1}) \left[\left(\bigwedge_{i=1}^{k+1} e_i \neq e \wedge \bigwedge_{i \neq j} e_i \neq e_j \right) \rightarrow \bigvee_{i=1}^{k+1} \neg \text{CROSSING}(E^*, e, e_i) \right] \right] \end{aligned}$$

Here we insist that G is Hamiltonian and that, for every edge e and any set of $k + 1$ distinct other edges, at least one among them does not cross e .

$$\begin{aligned} \text{CLOSED OUTER } k\text{-QUASI-PLANAR}_G &\equiv (\exists E^*) \left[\text{HAMILTONIAN}(E^*) \wedge \right. \\ &(\forall e_1, \dots, e_k) \left[\left(\bigwedge_{i \neq j} e_i \neq e_j \right) \rightarrow \bigvee_{i \neq j} \neg \text{CROSSING}(E^*, e_i, e_j) \right] \end{aligned}$$

Again, we insist that G is Hamiltonian and further that, for any set of k distinct edges, there is at least one pair among them that does not cross.

We conclude this section by mentioning an intermediate concept between closed outer k -planarity and outer k -planarity, i.e., *full outer k -planarity* [19]. The full outer k -planar graphs are defined as having a convex drawing which is k -planar and additionally there is no crossing on the outer boundary of the drawing. Hong and Nagamochi [19] gave a linear-time recognition algorithm for full outer 2-planar graphs. Clearly, the closed 2-planar graphs are a subclass of the full 2-planar graphs. So, one open question is whether one can generalize our MSO₂ expressions of closed outer k -planarity and closed outer k -quasi-planarity to the full versions. If yes, this would provide linear-time recognition of full outer k -planar graphs for every k , including the full outer 2-planar case.

Acknowledgement. We acknowledge Alexander Ravsky, Thomas van Dijk, Fabian Lipp, and Johannes Blum for their comments and preliminary discussion. We also thank David Wood for pointing us to references [13, 24, 30].

References

1. Ackerman, E.: On the maximum number of edges in topological graphs with no four pairwise crossing edges. *Discrete Comput. Geom.* **41**(3), 365–375 (2009). <https://doi.org/10.1007/s00454-009-9143-9>
2. Ackerman, E., Tardos, G.: On the maximum number of edges in quasi-planar graphs. *J. Combin. Theory Ser. A* **114**(3), 563–571 (2007). <https://doi.org/10.1016/j.jcta.2006.08.002>
3. Angelini, P., et al.: On the Relationship Between k -Planar and k -Quasi-Planar Graphs. In: Bodlaender, Hans L., Woeginger, Gerhard J. (eds.) WG 2017. LNCS, vol. 10520, pp. 59–74. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68705-6_5
4. Auer, C., Bachmaier, C., Brandenburg, F.J., Gleißner, A., Hanauer, K., Neuwirth, D., Reislhuber, J.: Outer 1-planar graphs. *Algorithmica* **74**(4), 1293–1320 (2016). <https://doi.org/10.1007/s00453-015-0002-1>
5. Babu, J., Khoury, A., Newman, I.: Every property of outerplanar graphs is testable. In: Jansen, K., Mathieu, C., Rolim, J.D.P., Umans, C. (eds.) APPROX/RANDOM 2016. LIPIcs, vol. 60, pp. 21:1–21:19. Schloss Dagstuhl, Leibniz-Zentrum für Informatik, Dagstuhl (2016). <https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2016.21>
6. Bannister, M.J., Eppstein, D.: Crossing minimization for 1-page and 2-page drawings of graphs with bounded treewidth. In: Duncan, C., Symvonis, A. (eds.) GD 2014. LNCS, vol. 8871, pp. 210–221. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45803-7_18
7. Binucci, C., Di Giacomo, E., Hossain, M.I., Liotta, G.: 1-page and 2-page drawings with bounded number of crossings per edge. In: Lipták, Z., Smyth, W.F. (eds.) IWOCA 2015. LNCS, vol. 9538, pp. 38–51. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29516-9_4
8. Borodin, O.V.: Solution of the Ringel problem on vertex-face coloring of planar graphs and coloring of 1-planar graphs. *Metody Diskret. Analiz.* **41**, 12–26, 108 (1984)
9. Capoleas, V., Pach, J.: A Turán-type theorem on chords of a convex polygon. *J. Combin. Theory Ser. B* **56**(1), 9–15 (1992). [https://doi.org/10.1016/0095-8956\(92\)90003-G](https://doi.org/10.1016/0095-8956(92)90003-G)
10. Courcelle, B.: The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inform. Comput.* **85**(1), 12–75 (1990). [https://doi.org/10.1016/0890-5401\(90\)90043-H](https://doi.org/10.1016/0890-5401(90)90043-H)
11. Courcelle, B., Engelfriet, J.: *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach*. Cambridge University Press (2012)
12. Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Lower bounds based on the exponential-time hypothesis. *Parameterized Algorithms*, pp. 467–521. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21275-3_14
13. Dress, A.W.M., Koolen, J.H., Moulton, V.: On line arrangements in the hyperbolic plane. *Eur. J. Comb.* **23**(5), 549–557 (2002). <https://doi.org/10.1006/eujc.2002.0582>

14. Dujmović, V., Eppstein, D., Wood, D.R.: Structure of graphs with locally restricted crossings. *SIAM J. Discrete Math.* **31**(2), 805–824 (2017)
15. Dvořák, Z., Norin, S.: Treewidth of graphs with balanced separations. ArXiv (2014). <http://arxiv.org/abs/1408.3869>
16. Fox, J., Pach, J., Suk, A.: The number of edges in k -quasi-planar graphs. *SIAM J. Discrete Math.* **27**(1), 550–561 (2013). <https://doi.org/10.1137/110858586>
17. Geneson, J., Khovanova, T., Tidor, J.: Convex geometric $(k + 2)$ -quasiplanar representations of semi-bar k -visibility graphs. *Discrete Math.* **331**, 83–88 (2014). <https://doi.org/10.1016/j.disc.2014.05.001>
18. Hong, S.H., Eades, P., Katoh, N., Liotta, G., Schweitzer, P., Suzuki, Y.: A linear-time algorithm for testing outer-1-planarity. *Algorithmica* **72**(4), 1033–1054 (2015). <https://doi.org/10.1007/s00453-014-9890-8>
19. Hong, S.-H., Nagamochi, H.: Testing full outer-2-planarity in linear time. In: Mayr, E.W. (ed.) *WG 2015. LNCS*, vol. 9224, pp. 406–421. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53174-7_29
20. Impagliazzo, R., Paturi, R.: On the complexity of k -SAT. *J. Comput. Syst. Sci.* **62**(2), 367–375 (2001). <https://doi.org/10.1006/jcss.2000.1727>
21. Kobourov, S.G., Liotta, G., Montecchiani, F.: An annotated bibliography on 1-planarity. ArXiv (2017). <http://arxiv.org/abs/1703.02261>
22. Lick, D.R., White, A.T.: k -degenerate graphs. *Canadian J. Math.* **22**, 1082–1096 (1970). <https://doi.org/10.4153/CJM-1970-125-1>
23. Masuda, S., Kashiwabara, T., Nakajima, K., Fujisawa, T.: On the NP-completeness of a computer network layout problem. In: *Proceedings of IEEE International Symposium Circuits and Systems*, pp. 292–295 (1987)
24. Nakamigawa, T.: A generalization of diagonal flips in a convex polygon. *Theor. Comput. Sci.* **235**(2), 271–282 (2000). [https://doi.org/10.1016/S0304-3975\(99\)00199-1](https://doi.org/10.1016/S0304-3975(99)00199-1)
25. Pach, J., Shahrokhi, F., Szegedy, M.: Applications of the crossing number. *Algorithmica* **16**(1), 111–117 (1996). <https://doi.org/10.1007/BF02086610>
26. Pach, J.: Graphs drawn with few crossings per edge. *Combinatorica* **17**(3), 427–439 (1997). <https://doi.org/10.1007/BF01215922>
27. Ringel, G.: Ein Sechsfarbenproblem auf der Kugel. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* **29**(1), 107–117 (1965). <https://doi.org/10.1007/BF02996313>
28. Robertson, N., Seymour, P.D.: Graph minors. III. Planar tree-width. *J. Combin. Theory Ser. B* **36**(1), 49–64 (1984). [https://doi.org/10.1016/0095-8956\(84\)90013-3](https://doi.org/10.1016/0095-8956(84)90013-3)
29. Schaefer, M.: The graph crossing number and its variants: a survey. *Electronic J. Combin. DS21*, 100 pages (2013, 2014). <http://www.combinatorics.org/ojs/index.php/eljc/article/view/DS21>
30. Wood, D.R., Telle, J.A.: Planar decompositions and the crossing number of graphs with an excluded minor. *New York J. Math.* **13**, 117–146 (2007)
31. Yannakakis, M.: Embedding planar graphs in four pages. *J. Comput. Syst. Sci.* **38**(1), 36–67 (1989). [https://doi.org/10.1016/0022-0000\(89\)90032-9](https://doi.org/10.1016/0022-0000(89)90032-9)

The Effect of Planarization on Width

David Eppstein^(✉)

Department of Computer Science, University of California, Irvine, USA
eppstein@uci.edu

Abstract. We study the effects of planarization (the construction of a planar diagram D from a non-planar graph G by replacing each crossing by a new vertex) on graph width parameters. We show that for treewidth, pathwidth, branchwidth, clique-width, and tree-depth there exists a family of n -vertex graphs with bounded parameter value, all of whose planarizations have parameter value $\Omega(n)$. However, for bandwidth, cutwidth, and carving width, every graph with bounded parameter value has a planarization of linear size whose parameter value remains bounded. The same is true for the treewidth, pathwidth, and branchwidth of graphs of bounded degree.

1 Introduction

Planarization is a graph transformation, standard in graph drawing, in which a given graph G is drawn in the plane with simple crossings of pairs of edges, and then each crossing of two edges in the drawing is replaced by a new dummy vertex, subdividing the two edges [1–4]. This should be distinguished from a different problem, also called planarization, in which we try to find a large planar subgraph of a nonplanar graph [5–8]. A given graph G may have many different planarizations, with different properties. Although the size of the planarization (equivalently the crossing number of G) is of primary importance in graph drawing, it is natural to ask what other properties can be transferred from G to its planarizations.

One problem of this type arose in the work of Jansen and Wulms on the fixed-parameter tractability of graph optimization problems on graphs of bounded pathwidth [9]. One of their constructions involved the planarization of a non-planar graph of bounded pathwidth, and they observed that the planarization maintained the low pathwidth of their graph. Following this observation, Jansen asked on csttheory.stackexchange.com whether planarization preserves the property of having bounded pathwidth, and in particular whether $K_{3,n}$ (a graph of bounded pathwidth) has a bounded-pathwidth planarization.¹ This paper represents an extended response to this problem. We provide a negative answer

Supported in part by the National Science Foundation under Grants CCF-1228639, CCF-1618301, and CCF-1616248. The author is grateful to Glencora Borradaile, Erin Chambers, and Amir Nayyeri for discussions that helped clarify the distinctions between some of the width parameters considered here.

¹ See <https://csttheory.stackexchange.com/q/35974/95>.

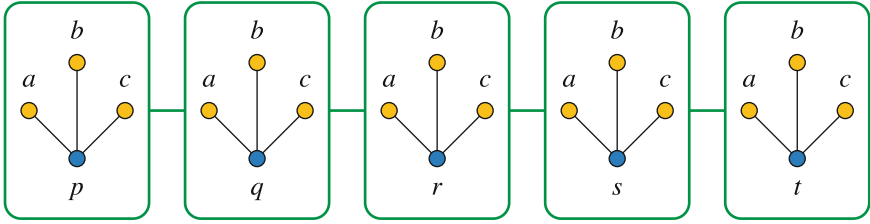


Fig. 1. A tree-decomposition and path-decomposition of $K_{3,5}$, with width three. Vertices a , b , and c (on one side of the bipartition) belong to all bags; vertices p , q , r , s , and t (on the other side) are each in only one bag.

to Jansen’s question: planarizations of $K_{3,n}$ do not have bounded pathwidth. However, for bounded-degree graphs of bounded pathwidth, there always exists a planarization that maintains bounded pathwidth. More generally we study similar questions for many other standard graph width parameters.

Our work should be distinguished from a much earlier line of research on planarization and width, in which constraints on the width of planar graphs are transferred in the other direction, to information about the graph being planarized. In particular, Leighton [1] used the facts that planar graphs have width at most proportional to the square root of their size, and that (for certain width parameters) planarization cannot decrease width, to show that when the original graph has high width it must have crossing number quadratic in its width. In our work, in contrast, we are assuming that the original graph has low width and we derive properties of its planarization from that assumption.

1.1 Width Parameters in Graphs

There has been a significant amount of research on graph width parameters and their algorithmic implications; see Nešetřil and Ossona de Mendez [10] for a more detailed survey. We briefly describe the parameters that we use here.

Treewidth. Treewidth has many equivalent definitions; the one we use is that the treewidth of a graph G is the minimum width of a tree-decomposition of G [10]. Here, a tree-decomposition is a tree T whose nodes, called *bags*, are labeled by sets of vertices of G . Each vertex of G must belong to the bags of a contiguous subtree of T , and for each edge of G there must exist a bag containing both endpoints of the edge. The width of the decomposition is one less than the maximum cardinality of the bags. Figure 1 shows such a decomposition for $K_{3,5}$.

Pathwidth. The pathwidth of a graph is the minimum width of a tree-decomposition whose tree is a path, as it is in Fig. 1 [10]. Equivalently the pathwidth equals the minimum vertex separation number of a linear arrangement of the vertices of G (an arrangement of the vertices into a linear sequence) [11]. Every linear arrangement of an n -vertex graph defines $n - 1$ cuts, that is,

$n - 1$ partitions of the vertices into a prefix of the sequence and a disjoint suffix of the sequence. The vertex separation number of a linear arrangement is the maximum, over these cuts, of the number of vertices in the prefix that have a neighbor in the suffix. From a linear arrangement one can construct a tree-decomposition in the form of a path, where the first bag on the path for each vertex v contains v together with all vertices that are earlier than v in the arrangement but that have v or a later vertex as a neighbor.

Cutwidth. The cutwidth of a graph equals the minimum edge separation number of a linear arrangement of the vertices of G [12]. The edge separation number of a linear arrangement is the maximum, over the prefix-suffix cuts of the arrangement, of the number of edges that cross the cut.

Bandwidth. The bandwidth of a graph equals the minimum span of a linear arrangement of the vertices of G [12]. The span of a linear arrangement is the maximum, over the edges of G , of the number of steps in the linear arrangement between the endpoints of the edge.

Branchwidth. A branch-decomposition of a graph G is an undirected tree T , with leaves labeled by the edges of G , and with every interior vertex of T having degree three. Removing any edge e from T partitions T into two subtrees; these subtrees partition the leaves of T into two sets, and correspondingly partition the edges of G into two subgraphs. The width of the decomposition is the maximum, over all edges e of T , of the number of vertices that belong to both subgraphs. The branchwidth of G is the minimum width of any branch-decomposition [13].

Carving width. A carving decomposition of a graph G is an undirected tree T , with leaves labeled by the vertices of G , and with every interior vertex of T having degree three. Removing any edge e from T partitions T into two subtrees; these subtrees partition the leaves of T into two sets, and correspondingly partition the vertices of G into two induced subgraphs. The width of the decomposition is the maximum, over all edges e of T , of the number of edges of G that connect one of these subgraphs to the other. The carving width of G is the minimum width of any carving decomposition [13]. For instance, Fig. 6 depicts a carving decomposition of $K_{3,3}$ with width four, the minimum possible for this graph.

Tree-depth. The tree-depth of G is the minimum depth of a depth-first-search tree T of a supergraph of G (Fig. 2). Such a tree can be characterized more simply by the property that every edge of G connects an ancestor-descendant pair in T [10].

Clique-width. A clique-construction of a graph G is a process that constructs a vertex-colored copy of G from smaller vertex-colored graphs by steps that create a new colored vertex, take the disjoint union of two colored graphs, add all edges from vertices of one color to vertices of another, or assigning a new color to vertices of a given color. The width of a clique-construction is the number of distinct colors it uses, and the clique-width of a graph is the minimum width of a clique-construction [14].

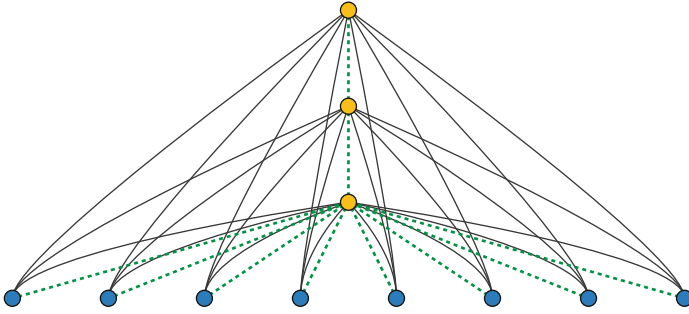


Fig. 2. $K_{3,8}$ has tree-depth three: The depth-three tree shown by the green dashed edges forms a depth-first search tree of a supergraph of $K_{3,8}$. (Color figure online)

1.2 New Results

In this paper, we consider for each of the depth parameters listed above how the parameter can change from a graph to its planarization, when the planarization is chosen to minimize the parameter value. We show that for treewidth, pathwidth, branchwidth, tree-depth, and clique-width there exists a graph with bounded parameter value, all of whose planarizations have parameter value $\Omega(n)$. In each of these cases, the graph can be chosen as a complete bipartite graph $K_{3,n}$. (It was also known that the planarizations of $K_{3,n}$ have quadratic size [15].)

However, for bandwidth, cutwidth, and carving width, every graph with bounded parameter value has a planarization of linear size whose parameter value remains bounded. The same is true for the treewidth, pathwidth, branchwidth, and clique-width of graphs of bounded degree. (Graphs of bounded degree and bounded tree-depth have bounded size, so this final case is not interesting.)

2 Treewidth, Branchwidth, Pathwidth, Tree-Depth, and Clique-Width

In this section we show that all planarizations of $K_{3,n}$ have high width. We begin by computing the crossing number of $K_{3,n}$. This is a special case of Turán’s brick factory problem of determining the crossing number of all complete bipartite graphs. For our results we need a variant of the crossing number, $\text{cr}_{\text{pair}}(G)$, defined as the minimum number of pairs of crossing edges (allowing edges to cross each other multiple times, but only counting a single crossing in each case) instead of the usual crossing number $\text{cr}(G)$ defined as the number of points where edges cross [16]. We bound this number by an adaptation of an argument from Kleitman [17], who credits it to Zarankiewicz [15].

Lemma 1.

$$\text{cr}_{\text{pair}}(K_{3,n}) = \binom{\lfloor n/2 \rfloor}{2} + \binom{\lceil n/2 \rceil}{2} = \left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n-1}{2} \right\rfloor.$$

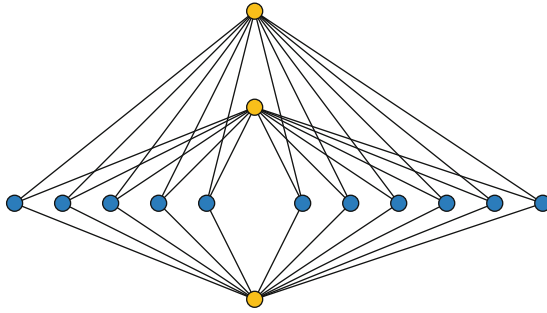


Fig. 3. A drawing of $K_{3,11}$ with 25 crossings, the minimum possible for this graph.

Proof. To show that a drawing with this many crossing pairs exists, place the n vertices on one side of the bipartition of $K_{3,n}$ along the x -axis, with $\lfloor n/2 \rfloor$ on one side of the origin and $\lceil n/2 \rceil$ on the other. Place the three vertices on the other side of the bipartition along the y -axis, with two points on one side of the origin and one on the other. Connect all of the pairs of points that have one point on each axis by a straight line segment, as shown in Fig. 3. A straightforward calculation shows that the number of crossings is as claimed.

In the other direction, we know as base cases that $\text{cr}_{\text{pair}}(K_{3,2}) = 0$ and $\text{cr}_{\text{pair}}(K_{3,3}) = 1$. For any larger n , let the vertices of the n -vertex side of the bipartition of $K_{3,n}$ be v_1, v_2, \dots, v_n . If every pair v_i, v_j form the endpoints of at least one pair of crossing edges, then the total number of crossings is at least $\binom{n}{2}$, larger than the stated bound; otherwise, order the vertices so that v_{n-1} and v_n do not form the endpoints of any pair of crossing edges.

Then, in any drawing of $K_{3,n}$, the $K_{3,n-2}$ subgraph formed by deleting v_{n-1} and v_n has $\text{cr}_{\text{pair}}(K_{3,n-2})$ crossings. Each of the $n - 2$ $K_{3,3}$ subgraphs induced by v_{n-1}, v_n , exactly one other v_i , and the three vertices on the other side of the bipartition supplies at least one additional crossing, because $\text{cr}_{\text{pair}}(K_{3,3}) = 1$. None of these subgraphs share any crossings, because the crossings in the $K_{3,n-2}$ subgraph involve neither v_{n-1} nor v_n , while the crossings in the $K_{3,3}$ subgraph all involve exactly one of these two vertices and the one other vertex v_i included in the subgraph. Therefore, we have that

$$\text{cr}_{\text{pair}}(K_{3,n}) \geq \text{cr}_{\text{pair}}(K_{3,n-2}) + (n - 2) \text{cr}_{\text{pair}}(K_{3,3}).$$

The result follows by induction on n . □

This lemma shows that the *crossing graph* of a drawing, with a vertex in the crossing graph for each edge of $K_{3,n}$ and an edge in the crossing graph for each crossing of the drawing, has constant density, in the following sense. We define the *density* of a graph with m edges and n vertices to be $m/\binom{n}{2}$. This is a number in the range $[0, 1]$. For instance, the crossing graph of any planarization of $K_{3,n}$ has $3n$ vertices and (by Lemma 1) at least $(1 - o(1))n^2/4$ edges, so its density is at least

$$(1 - o(1)) \frac{n^2}{4} / \binom{3n}{2} = \frac{1}{18} - o(1).$$

To prove that planarizations of $K_{3,n}$ have high treewidth, we need higher densities than this, which we will achieve using the following “densification lemma”:

Lemma 2. *Let G be a disconnected graph with n vertices and m edges, such that the i th connected component of G has n_i vertices and m_i edges. Then there exists i such that $m_i/n_i \geq m/n$.*

Proof. We can represent m/n as a convex combination of the corresponding quantities in the subgraphs:

$$\frac{m}{n} = \sum_i \frac{n_i}{n} \cdot \frac{m_i}{n_i}.$$

The result follows from the fact that a convex combination of numbers cannot exceed the maximum of the numbers. □

Given a tree-decomposition T of a drawing D of $K_{3,n}$, and any connected subtree S of T , we define a crossing graph C_S as follows. Let E_S be the subset of edges of $K_{3,n}$ with the property that, for each edge e in E_S , the only bags of T that contain crossings on e are the bags in S . (We do *not* require the two endpoints of e in $K_{3,n}$ to belong to these bags.) Then C_S is a graph having E_S as its vertex set, and having an edge for each pair of edges in E_S that cross in D . For instance, C_T is the crossing graph of the whole drawing, as defined earlier. We will use Lemma 2 to find subtrees S whose graphs C_S are more dense (relative to their numbers of vertices) than C_T . To do so, we use the separation properties of trees:

Lemma 3. *Let T be a tree-decomposition T of a drawing D of $K_{3,n}$, and suppose that T has width w . Let S be a subtree of T such that C_S has n_S vertices and m_S edges. Then there exists a bag $B \in S$ with the following properties:*

- The removal of B disconnects S into subtrees S_i .
- For subtree S_i , the corresponding crossing graph C_{S_i} has at most $n_S/2$ vertices.
- The total number of edges in all of the crossing graphs C_{S_i} for all of the subtrees S_i is at least $m_S - 2(w + 1)(n - 2)$.

Proof. We choose B arbitrarily, and then as long as it fails to meet the condition on the number of vertices in the graphs C_{S_i} we move B to the (unique) adjacent bag in which this condition is not met. After the move, the subtree containing the former location of B has at most $n_S/2$ vertices in C_{S_i} , because these vertices are disjoint from the ones in the large subtree before the move. Moving B also cannot increase the numbers of vertices in the crossing graphs of the subtrees formed from the partition of the large subtree, and it reduces the numbers of

bags in those subtrees. Therefore, this process must eventually terminate at a choice of B for which all crossing graphs have the stated number of vertices.

An edge e in C_S (representing a crossing between two edges f and f' of $K_{3,n}$) will belong to one of the C_{S_i} unless B contains a crossing point on f or on f' . B may contain at most $w + 1$ crossings of D , and each may eliminate at most $2(n - 2)$ edges of C_S (if it is a crossing of two edges in E_S and each has $n - 2$ other crossings). Therefore, the total number of edges in all of the crossing graphs C_{S_i} for all of the subtrees S_i is as stated. \square

Theorem 1. *Every planarization of $K_{3,n}$ has treewidth $\Omega(n)$.*

Proof. Let D be an arbitrary planarization of $K_{3,n}$, and let T be a minimum-width tree-decomposition of D . Let $\epsilon > 0$ be a constant to be determined later. We will show that T either has width at least ϵn , or it has a subtree S whose crossing graph C_S has density strictly greater than one. Since no graph (without repeated edges) can have density so high, the only possibility is that T has width at least $\epsilon n = \Omega(n)$.

To find S , for drawings whose width is at most ϵn , begin with $S = T$. Then, repeatedly use Lemma 3 to partition the current choice of subtree S into smaller subtrees, and then use Lemma 2 to find one of these subtrees that is dense. Each such step reduces the number of vertices in C_S by at least a factor of two, while also reducing the number of edges by approximately the same reduction factor (approximately because of the $O(\epsilon n^2)$ edges of the crossing graph that are eliminated by the application of Lemma 3). Therefore, each step increases the density of C_S by a factor of $2 - O(\epsilon)$. When $S = T$, the density is at least $1/18 - o(1)$, so after at most five steps the density is $(32 - O(\epsilon))(1/18 - o(1))$.

To complete the argument, we need only choose ϵ to be small enough so that this expression, $(32 - O(\epsilon))(1/18 - o(1))$, has a value exceeding one. \square

Corollary 1. *For every planarization of $K_{3,n}$, and every parameter in {pathwidth, cutwidth, bandwidth, branchwidth, carving width, tree-depth, clique-width}, the value of the parameter on the planarization is $\Omega(n)$. Therefore, there exists a family of graphs for which each of these parameters is bounded but for each planarization has linear parameter value.*

Proof. All of these parameters except clique-width are bounded from below by a linear function of the treewidth.

As with any complete bipartite graph, the clique-width of $K_{3,n-3}$ is two: it can be constructed from a disjoint union of single vertices of two colors, by adding edges between all bichromatic pairs of vertices (Fig. 4). The $\Omega(n)$ lower bound on clique-width follows from the facts that (as a planar graph) any planarization has no $K_{3,3}$ subgraph and that, for graphs with no $K_{t,t}$ subgraph, the treewidth is upper-bounded by a constant factor (depending on t) times the clique-width [18]. Equivalently, the clique-width of any planarization is lower-bounded by a constant times its treewidth, which by Theorem 1 is $\Omega(n)$. \square

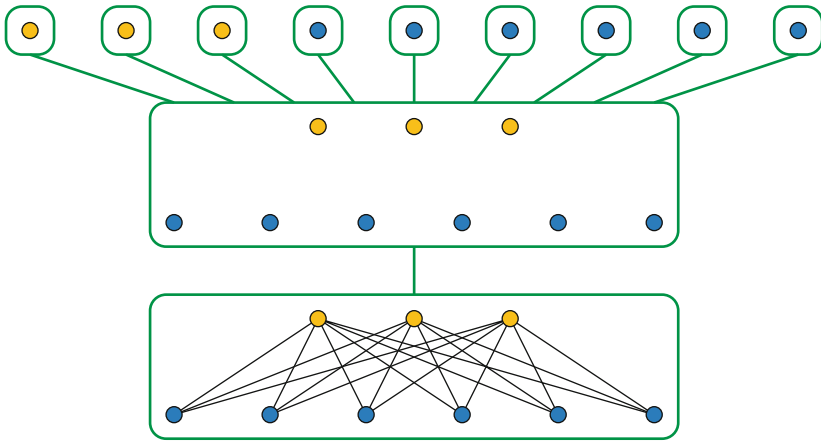


Fig. 4. Clique-width 2 construction of $K_{3,6}$ by a disjoint union of colored single vertices, followed by an operation that adds an edge between each bichromatic pair of vertices.

3 Cutwidth and Bounded-Degree Pathwidth

Cutwidth behaves particularly well under planarization:²

Theorem 2. *Let G be a graph with n vertices and m edges, of cutwidth w . Then G has a planarization with $O(n + wm)$ vertices, of cutwidth at most w .*

Proof. Consider a linear arrangement of G with edge separation number w , and use the positions in this arrangement as x -coordinates for the vertices. Assign the vertices y -coordinates that place them into convex and general position, draw the edges of G as straight line segments between the resulting points, and planarize the drawing by replacing each crossing by a vertex. Here, by “general position” we mean that no two points have the same x -coordinate, no five points form a pentagon in which two crossing points and a vertex have the same x -coordinate, no six points form a hexagon with three coincident diagonals, and no eight points form an octagon in which the crossing points of two pairs of diagonals have the same x -coordinate. This will all be true of a rotation by a sufficiently small but nonzero angle of any convex placement. In the resulting drawing, there can be no intersections of vertices or edges other than incidences and simple crossings, and no two vertices or crossing points can have the same x -coordinate. An example is shown in Fig. 5.

We use the ordering by x -coordinates of the planarization as a linear arrangement of the planarization. The edge intersection number is the maximum number of edges in the drawing that can be cut by any vertical line, unchanged between G and its planarization.

² After the appearance of the preprint version of this paper [19], we learned that this result has been obtained independently by van Geffen et al. [20].

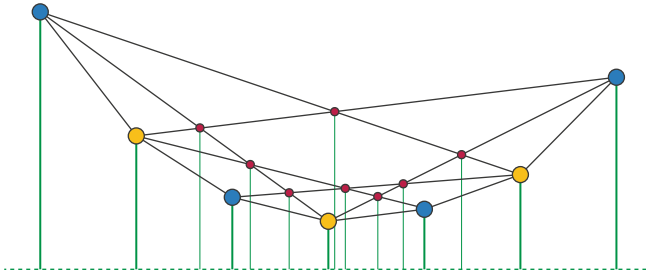


Fig. 5. Planarizing a graph of low cutwidth (here $K_{3,4}$, drawn with edge separation number six) by lifting its linear arrangement to a convex curve.

Because of the convex position of the vertices of G , each edge (u, v) of G can only be crossed by other edges that cross exactly one of the two vertical lines through u and v ; there are $O(w)$ such edges, so the number of crossings per edge is $O(w)$ and the total number of crossings is $O(wm)$. \square

The lower bound of Corollary 1 does not contradict Theorem 2 because $K_{3,n}$ does not have bounded cutwidth. Its cutwidth is at least $3\lceil n/2 \rceil$, obtained in any linear arrangement at the cut between the first $\lceil n/2 \rceil$ vertices on the n -vertex side of the bipartition (together with any vertices from the other side that are mixed among them) and the remaining vertices of the graph. For instance, the drawing of $K_{3,4}$ in Fig. 5 achieves the optimal cutwidth of six for this graph. An example showing that the planarization size bound is tight is given by a disjoint union of $O(n/w)$ bounded-degree expander graphs, each having $O(w)$ vertices and crossing number $\Theta(w^2)$.

Corollary 2. *Let G be a graph with bounded pathwidth and bounded maximum degree. Then G has a planarization with linear size and bounded pathwidth.*

Proof. If a graph has pathwidth w and maximum degree d , it has cutwidth at most dw [12], and so does its planarization (Theorem 2). Because the planarization has cutwidth at most dw , it also has pathwidth at most dw , because the vertex separation number of any linear arrangement is at most equal to the edge separation number (with equality when the separation number is achieved by a matching). \square

4 Bandwidth

The same construction used for planarizing graphs with low cutwidth also works for graphs of low bandwidth.

Theorem 3. *Let G be a graph with n vertices and m edges, of bandwidth w . Then G has a planarization with $O(n+w^2m)$ vertices, whose bandwidth is $O(w^4)$.*

Proof. We lift a linear arrangement of G with low span to a convex curve in the plane, as in the proof of Theorem 2. Within the span of any edge e of G , there are $O(w^2)$ other edges and $O(w^4)$ crossings of those edges, so the span of e in the planarization is $O(w^4)$. This bound applies also to the span of any segment of e created by crossings with other vertices. Each edge may be crossed by $O(w^2)$ other edges, so the total number of dummy vertices added is $O(w^2m)$. \square

It may be possible to reduce the bandwidth of the planarization by introducing artificial crossings to break up edges with long spans, but we have not pursued this approach as we do not believe it will lead to better graph drawings.

5 Carving Width and Bounded-Degree Treewidth

If a graph has low carving width, we can use its carving decomposition (a tree with the vertices at its leaves, internal degree three, and with few edges spanning the cut determined by each tree edge) to guide a drawing of the algorithm that leads to a planarization with low carving width.

It is helpful, for our construction, to relate carving width to cutwidth.

Lemma 4. *If a graph G has cutwidth w and maximum degree d , then G has carving width at most $\max(w, d)$.*

Proof. We form a carving decomposition of G in the form of a caterpillar: a path with each path vertex having a single leaf connected to it (except for the ends of the path which have two connected leaves). The ordering of the leaves is given by a linear arrangement minimizing the edge separation number. Then the cuts of the carving decomposition that are determined by edges of the path are exactly the ones determining the edge separation number, w . The remaining cuts, determined by leaf edges of the tree, are crossed by the neighboring edges of each vertex, of which there are at most d . An example of this construction can be seen in Fig. 5: the dashed horizontal green line represents the path from which the carving decomposition is formed, the heavy vertical green lines correspond to the leaf edges of the carving decomposition of $K_{3,4}$, and the thin vertical green edges correspond to the leaf edges of the carving decomposition of a planarization of $K_{3,4}$. \square

Theorem 4. *If an n -vertex graph G has carving width w , then G has a planarization with $O(w^2n)$ additional vertices that still has carving width at most w .*

Proof. Let T be the tree of a carving decomposition of G with width w . Draw T without crossings in the plane, with straight-line edges, and thicken the vertices of T to disks and the edges of T to rectangles without introducing any additional self-intersections of the drawing. Place each vertex of G in the disk of the corresponding leaf vertex of T . Route each edge of G as a curve through the rectangles and disks connecting its endpoints, so that within each rectangle it forms a monotone curve (with respect to the orientation of the rectangle) crossing at

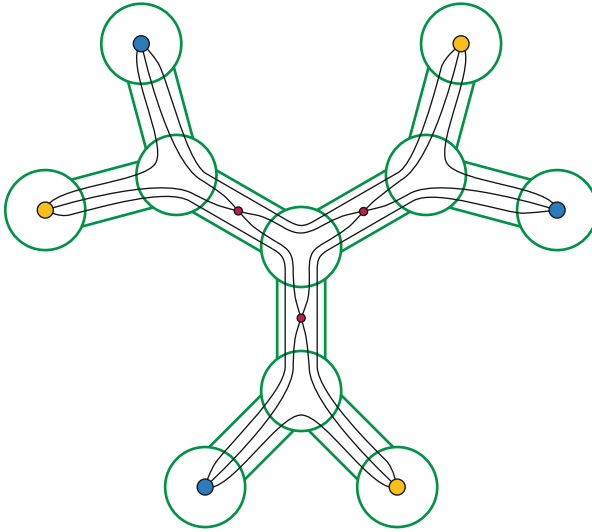


Fig. 6. Using a carving decomposition of $K_{3,3}$ to guide a planarization.

most once each other edge routed within the same rectangle, and so that, at each end of each rectangle, the curves are sorted by the ordering of their destination leaves in the planar embedding of T . With this sorted ordering, there need not be any crossings within the disks representing internal vertices of T , nor in the rectangles representing leaf edges of T (Fig. 6). The $n - 3$ remaining edges of T each contain at most $\binom{w}{2}$ crossings. So the total number of crossings is at most $(n - 3)\binom{w}{2} = O(w^2n)$.

This drawing cannot yet be recognized as a carving decomposition of a planarization of G , because some of its vertices (the dummy vertices introduced at crossings) are now placed along the edges of T rather than at leaves. However, by topologically sweeping the arrangement of monotone curves [21] within each rectangle corresponding to an edge of T , we can arrange the crossing points within that rectangle into a linear sequence, such that the portion of the drawing within that rectangle has edge separation number at most w for that sequence. Applying Lemma 4 (replacing the edge of T by a carving decomposition in the form of a caterpillar, with a leaf of the decomposition for each vertex added in the planarization to replace a crossing of G , and with the ordering of these leaves given by a topological sweep of the arrangement) produces a carving decomposition of the planarization with width w , as required. \square

We note that this planarization technique resembles the “simple planarization” method of Di Battista et al. [2] for clustered graphs. In this respect, we may view the carving decomposition of G as a clustering to be respected by the planarization.

In an appendix to the preprint version of this paper [19], we prove the following strengthening of Theorem 4:

Theorem 5. *If an n -vertex graph G has carving width w , then G has a planarization with $O(w^{3/2}n)$ additional vertices that still has carving width $O(w)$.*

An example showing that Theorem 5 is tight is given by a cluster graph consisting of $O(n/\sqrt{w})$ disjoint cliques of size $O(\sqrt{w})$, each requiring $\Theta(w^2)$ crossings in any drawing.

Corollary 3. *Let G be a graph with bounded treewidth or branchwidth and bounded maximum degree. Then G has a planarization with linear size and bounded treewidth and branchwidth.*

Proof. Treewidth and branchwidth are always within a constant factor of each other [13] so we may concentrate on the results for branchwidth, and the corresponding results for treewidth will follow automatically.

A carving decomposition may be converted into a branch decomposition by replacing each leaf of the carving decomposition (representing a vertex of the given graph) with a subtree (representing edges adjacent to the given vertex), in such a way that each edge is represented at exactly one of its endpoints. This increases the width of the decomposition by at most a factor equal to the degree. In the other direction, a branch decomposition may be converted into a carving decomposition by replacing each leaf of the branch decomposition (representing an edge of the given graph) by a subtree of zero, one, or two leaves (representing endpoints of the edge) in such a way that each vertex is represented at exactly one of its incident edges. This increases the width of the decomposition by at most a factor of two. So, the carving width is at most the degree times the branchwidth, and at least half the branchwidth [22].

Therefore, if G has bounded branchwidth and bounded maximum degree, it has bounded carving width, and Theorem 4 tells us that it has a planarization of linear size that also has bounded carving width. The same planarization also must have bounded branchwidth. \square

References

1. Leighton, F.T.: New lower bound techniques for VLSI. In: Proceedings of 22nd Symposium on Foundations of Computer Science (FOCS 1981), pp. 1–12. IEEE (1981)
2. Di Battista, G., Didimo, W., Marcandalli, A.: Planarization of clustered graphs. In: Mutzel, P., Jünger, M., Leipert, S. (eds.) GD 2001. LNCS, vol. 2265, pp. 60–74. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45848-4_5
3. Buchheim, C., Chimani, M., Gutwenger, C., Jünger, M., Mutzel, P.: Crossings and planarization. In: Tamassia, R. (ed.) Handbook of Graph Drawing and Visualization. Discrete Mathematics and its Applications, pp. 43–86. CRC Press (2014)
4. Garfunkel, S., Shank, H.: On the undecidability of finite planar graphs. *J. Symb. Log.* **36**, 121–126 (1971)
5. Thulasiraman, K., Jayakumar, R., Swamy, M.N.S.: On maximal planarization of nonplanar graphs. *IEEE Trans. Circ. Syst.* **33**(8), 843–844 (1986)
6. Cimikowski, R.: An analysis of heuristics for graph planarization. *J. Inform. Optim. Sci.* **18**(1), 49–73 (1997)

7. Chuzhoy, J., Makarychev, Y., Sidiropoulos, A.: On graph crossing number and edge planarization. In: Randall, D. (ed.) Proceedings of 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA 2011). Society for Industrial and Applied Mathematics, pp. 1050–1069 (2011)
8. Borradaile, G., Eppstein, D., Zhu, P.: Planar induced subgraphs of sparse graphs. *J. Graph Algorithms Appl.* **19**(1), 281–297 (2015)
9. Jansen, B.M.P., Wulms, J.J.H.M.: Lower bounds for protrusion replacement by counting equivalence classes. In: Guo, J., Hermelin, D. (eds.) Proceedings of 11th International Symposium on Parameterized and Exact Computation (IPEC 2016). Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl-Leibniz-Zentrum für Informatik, vol. 63, pp. 17:1–17:12 (2016)
10. Nešetřil, J., Ossona de Mendez, P.: Sparsity. Graphs, Structures, and Algorithms. AC, vol. 28. Springer, Heidelberg (2012). <https://doi.org/10.1007/978-3-642-27875-4>
11. Kinnersley, N.G.: The vertex separation number of a graph equals its path-width. *Inform. Process. Lett.* **42**(6), 345–350 (1992)
12. Chung, F.R.K., Seymour, P.D.: Graphs with small bandwidth and cutwidth. *Discrete Math.* **75**(1–3), 113–119 (1989)
13. Seymour, P.D., Thomas, R.: Call routing and the ratcatcher. *Combinatorica* **14**(2), 217–241 (1994)
14. Corneil, D.G., Rotics, U.: On the relationship between clique-width and treewidth. *SIAM J. Comput.* **34**(4), 825–847 (2005)
15. Zarankiewicz, K.: On a problem of P. Turan concerning graphs. *Fund. Math.* **41**, 137–145 (1954)
16. Pach, J., Tóth, G.: Which crossing number is it, anyway? In: Motwani, R. (ed.) Proceedings of 39th Symposium on Foundations of Computer Science (FOCS 1998), pp. 617–626. IEEE (1998)
17. Kleitman, D.J.: The crossing number of $K_{5,n}$. *J. Comb. Theory* **9**, 315–323 (1970)
18. Gurski, F., Wanke, E.: The tree-width of clique-width bounded graphs without $K_{n,n}$. In: Brandes, U., Wagner, D. (eds.) WG 2000. LNCS, vol. 1928, pp. 196–205. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-40064-8_19
19. Eppstein, D.: The effect of planarization on width. Electronic preprint [arxiv:1708.05155](https://arxiv.org/abs/1708.05155) (2017)
20. van Geffen, B.A.M., Jansen, B.M.P., de Kroon, N.A.W.M., Morel, R., Nederlof, J.: Optimal algorithms on graphs of bounded width (and degree): cutwidth sometimes beats treewidth, but planarity does not help. Manuscript (2017)
21. Edelsbrunner, H., Guibas, L.J.: Topologically sweeping an arrangement. *J. Comput. Syst. Sci.* **38**(1), 165–194 (1989)
22. Nestoridis, N.V., Thilikos, D.M.: Square roots of minor closed graph classes. *Discrete Appl. Math.* **168**, 34–39 (2014)

Contest Report

Graph Drawing Contest Report

William Devanny¹, Philipp Kindermann², Maarten Löffler³(✉),
and Ignaz Rutter⁴

¹ University of California, Irvine, USA
levnach@microsoft.com

² FernUniversität in Hagen, Hagen, Germany
philipp.kindermann@fernuni-hagen.de

³ Utrecht University, Utrecht, The Netherlands
m.loffler@uu.nl

⁴ Karlsruhe Institute of Technology, Karlsruhe, Germany
rutter@kit.edu

Abstract. This report describes the 24th Annual Graph Drawing Contest, held in conjunction with the 25th International Symposium on Graph Drawing (GD'17) in Boston, United States of America. The purpose of the contest is to monitor and challenge the current state of the art in graph-drawing technology.

1 Introduction

This year, the Graph Drawing Contest was divided into two parts: the *creative topics* and the *live challenge*.

The creative topics had two graphs: the first one was a graph about citations among previously presented papers in the Graph Drawing Symposium, and the second one described human metabolism and was previously one of the largest manually drawn graphs in biology. The data sets for the creative topics were published almost a year in advance, and contestants could solve and submit their results before the conference started. The submitted drawings were evaluated according to aesthetic appearance, domain-specific requirements, and how well the data was visually represented.

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. This year's topic was to maximize the smallest crossing angle in a straight-line drawing of a graph with vertex locations restricted to a grid.

Overall, we received 12 submissions: 3 submissions for the creative topics and 9 submissions for the live challenge.

2 Creative Topics

The two creative topics for this year were a graph about the graph drawing citations, and a human metabolism hypergraph. The goal was to visualize each

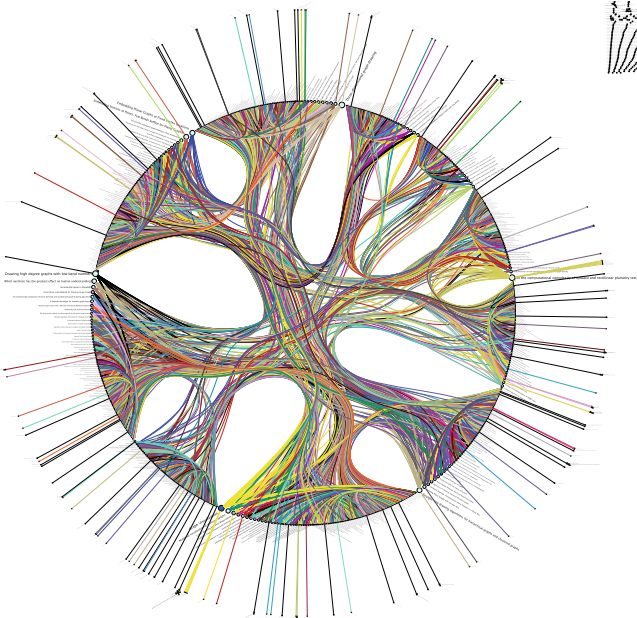
graph with complete artistic freedom, and with the aim of communicating the data in the graph as well as possible.

We received 2 submissions for the first topic, and 1 for the second. For each topic, we selected up to three contenders for the prize, which were printed on large poster boards and presented at the Graph Drawing Symposium. Finally, out of those contenders, we selected the winning submission. We will now review the top three submissions for each topic (for a complete list of submissions, refer to <http://www.graphdrawing.de/contest2017/results.html>).

2.1 Graph Drawing Citations

For the first creative topic, we extracted data of all publications in the Proceedings of Graph Drawing between 1994 and 2015. For each paper, we provide the title, the authors, the institutions of the authors, the year, and the citations. Your task is to create a nice visualization of the given citation network. It is desirable but not compulsory to make use of the extra data (authors, institutions, years) or a subset of it to explore interesting structures in the network.

Runner-up: Da Ye (University of Sydney). The committee likes the combination of clustering with radial layouts and organic edges, and a circular layout for the clusters that are connected with bundled edges. The visualization gives a good global overview of the graph structure, showing high-level connections between different topics in the community.

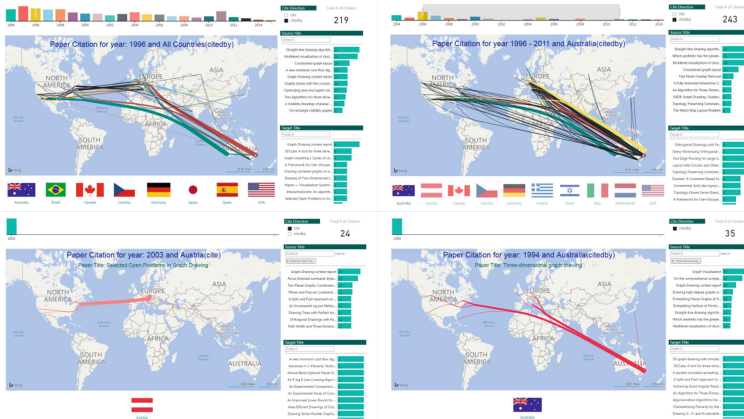


Winner: Steven Shangzhou Wang (University of Sydney). The committee likes the geolocated vertex placement in this submission, which gives a good picture of the local and global collaborations in the community. The interactive visualization allows the user to focus only on citations from certain papers, countries, or years, making it possible to explore the global graph structure as well as zoom in on specific details.

The graph can be explored here http://www.msbicoe.com/misc/graph_citation.html.

“ The location-based citation network can easily identify where the papers were cited by (or citing based on the cite direction selection). The directed flow from source to target paper is converged at the target, the thickness of the edge indicates how many times the source paper was cited by. The context menu in the interactive view displays more detailed information like authors who contributed to the paper. ”

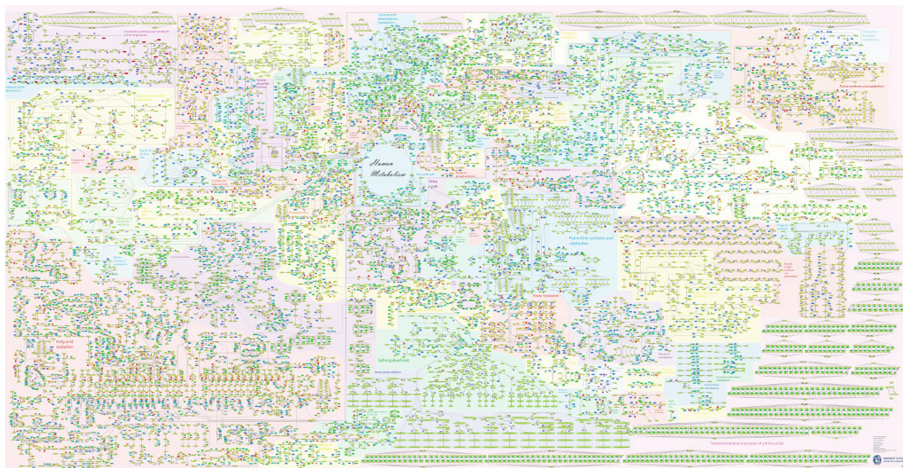
Steven Wang



2.2 Human Metabolism

Recon 2 is a community-driven, consensus ‘metabolic reconstruction’, which is the most comprehensive representation of human metabolism that is applicable to computational modeling. It was published by Thiele et al. in *Nature Biotechnology* [2]. Recon 2 can be used to identify causes of and new treatments for diseases like cancer, diabetes and even psychiatric and neurodegenerative disorders. Each person’s metabolism, which represents the conversion of food sources into energy and the assembly of molecules, is determined by genetics, environment and nutrition. Metabolic imbalances is an underlying cause of disease. Recon 2 merges complex details into a single interactive map and allows researchers to use this existing gene expression data and knowledge of the entire metabolic network to, for example, figure out how certain drugs would affect specific metabolic pathways found to create the conditions for cancerous cell growth.

ReconMap 2.0 is an interactive visualization of Recon 2 by Noronha et al. [1]. The map was drawn manually by 5 undergraduate biochemists over a total of 20 months and is, to the best of our knowledge, the largest manually drawn hypergraph in biology. A Constraint-Based Reconstruction and Analysis (COBRA) Toolbox extension to interact with ReconMap is available at <https://github.com/opencobra/cobratoolbox>, and the map can also be explored at <http://vmh.uni.lu/#mapnavigator>. There are 7440 metabolite reactions (hyperedges) and 5063 metabolite (nodes).



Winner: Fabian Klute and Irene Parada (TU Wien and TU Graz).

The committee likes the organic look of the submission; the use of curved and mostly short edges makes it possible to follow individual connections even in the somewhat chaotic center of the drawings. The committee also likes the way in which the visualization managed to cluster and label distinct sections of the graph. The labels of individual nodes are somewhat hard to read, but the use of colour partially alleviates that problem. The committee believes that the approach taken by the authors could serve as a starting point for an alternative visualization of Recon 2 that could be used more effectively by domain experts.

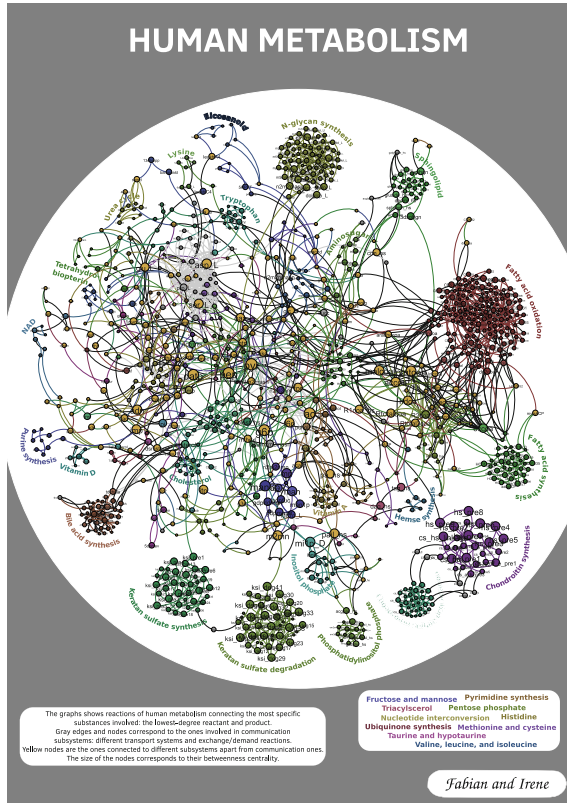
“ In order to filter the information, we thin out the graph in a preprocessing step. For every reaction we keep only the lowest-degree reactant and product. Isolated and degree one nodes are removed and only the biggest component is kept. Additionally we identify nodes corresponding to different biological compartments (e.g. $\text{co2}[\text{c}]$, $\text{co2}[\text{e}]$ are contracted into one co2 node).

To generate the layout we add edges to keep the subsystems closer together. The size of the nodes depends on its betweenness in the reduced graph. Then the layout is calculated using the Fruchterman-Reingold algorithm. Finally we color the nodes based on subsystems. We distinguish two big groups

of subsystems, the *communication* or *transportation* ones and the *proper* ones. As the name suggests the first kind mainly connects the proper subsystems. Based on this we categorize nodes into five cases, depending on if they are inside a proper subsystem, on its boundary, in a transport system, on the boundary between two different transport systems, or between two proper subsystems.

Fabian Klute

”



3 Live Challenge

The live challenge took place during the conference and lasted exactly one hour. During this hour, local participants of the conference could take part in the manual category (in which they could attempt to solve the graphs using a supplied tool), or in the automatic category (in which they could use their own software to solve the graphs). At the same time, remote participants could also take part in the automatic category.

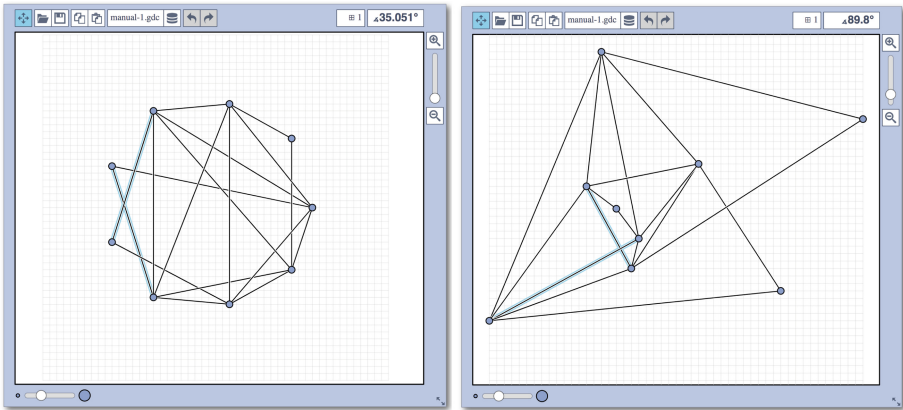
The challenge focused on maximizing the minimum crossing angle in a straight-line embedding of a given graph, with vertex locations restricted to a grid. The results were judged solely with respect to the minimum crossing angle;

other aesthetic criteria were not taken into account. This allows an objective way to evaluate each drawing.

From the resulting drawings, the committee does conclude that, if the minimum crossing angle is an indicator of the legibility of a graph visualization, it must be so only when combined with other criteria; for instance, penalizing low distances between vertices or vertices and non-adjacent edges.

3.1 Manual Category

In the manual category, participants were presented with seven graphs. These were arranged from small to large and chosen to highlight different types of graphs and graph structures. For illustration, we include the first graph in its initial state and the best manual solution we received (by team Aaaaaaah). For the complete set of graphs and submissions, refer to the contest website.



We are happy to present the full list of scores for all teams. The numbers listed are the smallest crossing angle in degrees in each graph; the horizontal bars visualize the corresponding scores.

graph	1	2	3	4	5	6	7
cold	49°	58°	90°	43°	41°	76°	36°
Aaaaaaah	90°	57°	46°	47°	46°	67°	38°
GraphSolvers	71°	57°	46°	46°	29°	37°	32°
🧑	88°	40°	85°	40°	35°	56°	23°
1337	89°	58°	78°	43°	47°	56°	17°

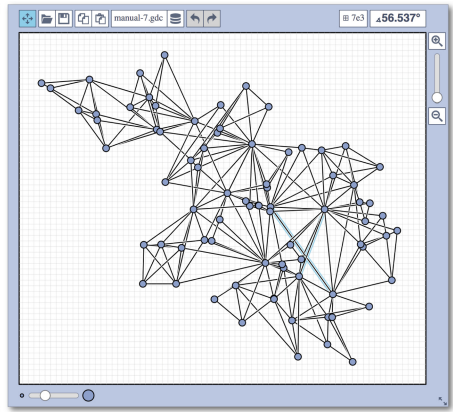
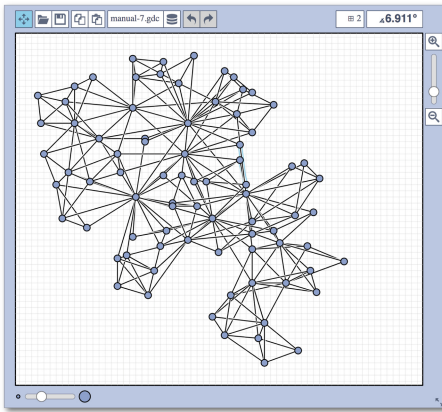
The winning team is team Aaaaaaah, consisting of Theresa Fröschl, Jonathan Klawitter, and Darren Strash!

“ We entered the contest in a state of total panic and excitement (see team name). Without any conscious strategy we tackled graph after graph, often removing one bad angle at a time. To comfort ourselves with the (false) belief that we actually had a strategy, we also tried simple global optimizations: bundling non-crossing edges in parallel, shrinking planar subgraphs so edges would not cross them, and moving badly-behaving vertices far away from the others (who knows, they might be contagious). Constantly in fear of destroying our progress with just the next step, which happened often, we celebrated each minor improvement with a save and a submission of our current best solution. ”

Jonathan Klawitter

3.2 Automatic Category

In the automatic category, participants had to solve the same seven graphs as in the manual category, and in addition another nine larger graphs. Again, the graphs were constructed to have different structure. Once more, for illustration, we include the best solution (by team CoffeeVM) of the first large graph. The graphs themselves can be found on the contest website.



graph 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Oh-Hyun Kwon	79°	59°	90°	50°	40°	84°	31°	45°	28°	7°	14°	11°	7°	5°	6°	2°
Arizona Anglers	48°	30°	78°	40°	28°	36°	3°	1°	7°	4°	5°	0°	2°	2°	0°	0°
TuebingenMidnight	77°	42°	89°	89°	30°	78°	34°	61°	9°	4°	6°	5°	4°	5°	4°	3°
CoffeeVM	90°	88°	90°	90°	80°	90°	57°	85°	60°	21°	47°	36°	25°	34°	21°	20°

The winning team is team CoffeeVM, consisting of Almut Demel, Dominik Dürschnabel, Tamara Mchedlidze, Marcel Radermacher and Lasse Wulf!

“ We use a two-step procedure. In the first step we apply energy-based approach modified to increase the crossing angles. In the second step we improve the result by a local search procedure. For larger graphs, we apply a heuristic to speed up the calculations of the crossing angles. ”
Lasse Wulf

Acknowledgments. The contest committee would like to thank the generous sponsors of the symposium, Dennis van der Wals for programming most of the tool for the manual category, and all the contestants for their participation. Further details including all submitted drawings and challenge graphs can be found at the contest website: <http://www.graphdrawing.de/contest2017/results.html>.

References

1. Noronha, A., Daniëlsdóttir, A.D., Jóhannsson, P.G.F., Jónsdóttir, S., Gunnarsson, S.J.J.P., Brynjólfsson, S., Schneider, R., Thiele, I., Fleming, R.M.T.: Reconmap: an interactive visualization of human metabolism. *Bioinformatics* **33**(4), 605–607 (2016)
2. Thiele, I., Swainston, N., Fleming, R.M.T., Hoppe, A., Sahoo, S., Aurich, M.K., Haraldsdóttir, H., Mo, M.L., Rolfsson, O., Stobbe, M.D., Thorleifsson, S.G., Agren, R., Bölling, C., Bordel, S., Chavali, A.K., Dobson, P., Dunn, W.B., Endler, L., Hala, D., Hucka, M., Hull, D., Jameson, D., Jamshidi, N., Jonsson, J.J., Juty, N., Keating, S., Nookaew, I., Novère, N.L., Malys, N., Mazein, A., Papin, J.A., Price, N.D., Selkov Sr., E., Sigurdsson, M.I., Simeonidis, E., Sonnenschein, N., Smallbone, K., Sorokin, A., van Beek, J.H.G.M., Weichart, D., Goryanin, I., Nielsen, J., Westerhoff, H.V., Kell, D.B., Mendes, P., Palsson, B.Ø.: A community-driven global reconstruction of human metabolism. *Nat. Biotechnol.* **31**, 419–425 (2013)

Poster Abstracts

Minimizing Wiggles in Storyline Visualizations

Theresa Fröschl and Martin Nöllenburg^(✉)

Algorithms and Complexity Group, TU Wien, Vienna, Austria
noellenburg@ac.tuwien.ac.at

A storyline visualization is a two-dimensional drawing of a special kind of time-varying hypergraph $H(t)$, where the x-axis represents time and the vertices (also called *characters*) are x-monotone curves. At each point in time t , the vertices form a permutation such that groups of adjacent characters in $H(t)$ occupy consecutive vertical positions to indicate a *meeting* at time t , see Fig. 1. Each character can only be part of at most one meeting at each point in time. This kind of visualization has been introduced for illustrating movie narratives [8], but is also more generally used in information visualization [6, 11].

Several aesthetic optimization criteria have been proposed [6, 11], including minimization of crossing, line wiggles, and white-space gaps. While crossing minimization has been studied from an algorithmic point of view in recent years [4, 5, 7], minimizing line wiggles, as another important quality criterion, which is similar to bend minimization in node-link diagrams [9, 10], has not been investigated on its own. We note that the problem of minimizing corners or moves in permutation diagrams [2, 3] is related to wiggle minimization, yet does not include the temporal aspects of storylines with meetings over time and their induced character ordering constraints. We present the first integer linear programming (ILP) model for exact wiggle minimization in storyline visualizations without an initial permutation. We can include crossing minimization into a weighted multicriteria ILP model and show examples of a first case study.

ILP formulation. A storyline visualization can be encoded as an $m \times p$ matrix with columns for the p time points, where meetings start or end, and $m > n$ rows for the slots used by the n characters of $H(t)$, where m is chosen large enough to allow for blank lines between different meetings at the same time points. The position of character i at time point t is expressed as a binary variable $x_{i,j}^t$ that is set to 1 if and only if i uses slot j at time point t . No two characters can use the same slot at the same time point ($\sum_{i=1}^n x_{i,j}^t \leq 1$). With this information about the position of the characters over time it is possible to determine the line wiggles of a character i by comparing the position of i for two successive time points t and $t + 1$. If the position changes, a wiggle is detected. The absolute value of the difference of the occupied slots at both time points yields the *height* of the wiggle, which is identified with the variable z_i^t . Using this height as the weight of a wiggle we get the following ILP model with the total wiggle height as objective

$$\text{minimize } \sum_{i=1}^n \sum_{t=1}^{p-1} z_i^t$$

$$\text{subject to } \sum_{j=1}^m j \cdot (x_{i,j}^t - x_{i,j}^{t+1}) \leq z_i^t, \quad \sum_{j=1}^m j \cdot (x_{i,j}^{t+1} - x_{i,j}^t) \leq z_i^t \quad \text{for all } z_i^t.$$

In addition we need to define constraints for correctly representing the character meetings and, for better visual distinction, keeping a blank line between any neighboring characters that do not meet. For a meeting e with k members between time points t_0 and t_1 we define integer variables $j_{\min}^{e,t}$ and $j_{\max}^{e,t}$ for the minimum and maximum slots for e at time points t with $t_0 \leq t \leq t_1$. The difference of these two slots must be exactly $j_{\max}^{e,t} - j_{\min}^{e,t} = k - 1$. By comparing the variables $j_{\min}^{e,t}$ and $j_{\max}^{e',t}$ for distinct meetings e and e' at the same time point t it is possible to define constraints that require blank lines between e and e' .

Finally, by using the position variables of any two characters a and b a binary comparison variable $y_{a,b}^t$ for this pair of characters can be created which takes value 1 if and only if a is placed above b at time point t by the constraints

$$m \cdot y_{a,b}^t \geq \sum_{j=1}^m j \cdot x_{b,j}^t - \sum_{j=1}^m j \cdot x_{a,j}^t, \quad y_{a,b}^t + y_{b,a}^t = 1.$$

A crossing between the characters a and b at time point t can be determined if the equation $y_{a,b}^t + y_{a,b}^{t+1} = 1$ is satisfied. If there is no crossing it evaluates to 0 or 2. With this a secondary objective function for crossing minimization can be added to the ILP, similar to the crossing minimization of Gronemann et al. [4].

Implementation. The ILP model is implemented using Gurobi [1]. Figure 1 illustrates results of a snippet of the movie Inception. Figure 1a shows the result of pure wiggle minimization with a total wiggle height of 55 (and 26 crossings) found after 33.37 min, while Fig. 1b shows the result of minimizing wiggles and crossings in a weighted multi-objective way with weight 1 for wiggles and weight 3 for crossings; it has a total wiggle height of 59 and 20 crossings and was found after 16.65 min. The multi-objective seems to produce more appealing results, yet there is still a need for improvements, especially for larger instances. We finally note that the ILP can be modified to minimize the number of wiggles or the maximum wiggle height instead of the total wiggle height.

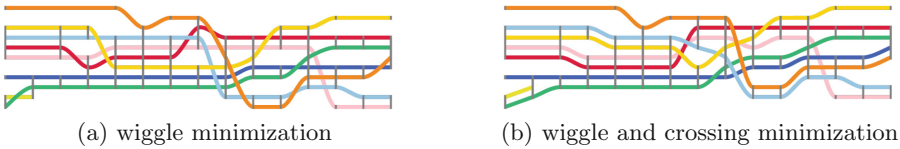


Fig. 1. Example snippets of storyline visualizations for the movie Inception; meetings are indicated by vertical lines

References

1. Gurobi optimizer 7.5 (2017). <http://www.gurobi.com>. Accessed 3 Aug 2017
2. Bereg, S., Holroyd, A.E., Nachmanson, L., Pupyrev, S.: Drawing permutations with few corners. In: Wismath, S., Wolff, A. (eds.) GD 2013. LNCS, vol. 8242, pp. 484–495. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03841-4_42
3. Bereg, S., Holroyd, A.E., Nachmanson, L., Pupyrev, S.: Representing permutations with few moves (2015). arXiv preprint [arXiv:1508.03674](https://arxiv.org/abs/1508.03674)
4. Gronemann, M., Jünger, M., Liers, F., Mambelli, F.: Crossing minimization in storyline visualization. In: Hu, Y., Nöllenburg, M. (eds.) GD 2016. LNCS, vol. 9801, pp. 367–381. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_29
5. Kostitsyna, I., Nöllenburg, M., Polishchuk, V., Schulz, A., Strash, D.: On minimizing crossings in storyline visualizations. In: Di Giacomo, E., Lubiw, A. (eds.) GD 2015. LNCS, vol. 9411, pp. 192–198. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27261-0_16
6. Liu, S., Wu, Y., Wei, E., Liu, M., Liu, Y.: Storyflow: tracking the evolution of stories. *IEEE Trans. Vis. Comput. Graph.* **19**(12), 2436–2445 (2013)
7. van Dijk, T.C., Fink, M., Fischer, N., Lipp, F., Markfelder, P., Ravsky, A., Suri, S., Wolff, A.: Block crossings in storyline visualizations. In: Hu, Y., Nöllenburg, M. (eds.) GD 2016. LNCS, vol. 9801, pp. 382–398. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_30
8. Munroe, R.: *Xkcd# 657: Movie narrative charts* (2009)
9. Purchase, H.: Which aesthetic has the greatest effect on human understanding? In: Di Battista, G. (ed.) GD 1997. LNCS, vol. 1353, pp. 248–261. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63938-1_67
10. Purchase, H.C., Cohen, R.F., James, M.: Validating graph drawing aesthetics. In: Brandenburg, F.J. (ed.) GD 1995. LNCS, vol. 1027, pp. 435–446. Springer, Heidelberg (1996). <https://doi.org/10.1007/BFb0021827>
11. Tanahashi, Y., Ma, K.L.: Design considerations for optimizing storyline visualizations. *IEEE Trans. Vis. Comput. Graph.* **18**(12), 2679–2688 (2012)

Graph Drawing for Formalized Diagrammatic Proofs in Geometry

Nathaniel Miller^(✉)

University of Northern Colorado, Greeley, USA
nathaniel.miller@unco.edu

CDEG, “Computerized Diagrammatic Euclidean Geometry,” is a computerized formal system for giving diagrammatic proofs in Euclidean geometry which uses planar graphs in drawing its diagrams. Here we discuss some of the graph-theoretic problems that arise in this context. This computer proof system implements a diagrammatic formal system for giving diagram-based proofs of theorems of Euclidean geometry that are similar to the informal proofs found in Euclid’s *Elements*. The theoretical ideas underlying this system and the original version of **CDEG** are described in detail in the book *Euclid and his Twentieth Century Rivals: Diagrams in the Logic of Euclidean Geometry* [4]. A much updated version of **CDEG** is now publicly available at [1]. Interested readers are encouraged to download **CDEG** and to try it out for themselves.

When we say that **CDEG** is a diagrammatic computer proof system, this means that it allows its user to give geometric proofs using diagrams. Internally, **CDEG** represents a diagram abstractly as a planar graph along with some additional information about how elements of the graph relate to the geometric objects they represent. The nodes in the graph represent points in the plane, while the edges represent line segments and arcs of circles. In general, one diagram drawn by **CDEG** can actually represent many different possible collections of lines and circles in the plane. What these collections all share, and share with the diagram that represents them, is that they all have the same planar topology. This means that any one can be stretched into any other. So, for example, a diagram containing a single line segment represents all possible single line segments in the plane, since any such line segment can be stretched into any other.

Two sample **CDEG** diagrams are shown in Fig. 1. The first diagram represents a circle drawn with a point inside of it and two points on its circumference. The circle is drawn in a way that appears to be rectangular rather than circular, but recall that all we care about here is the topology of the diagram. The second diagram, which occurs in the proof of Euclid’s Proposition 1, shows an equilateral triangle inscribed in the intersection of two circles. In **CDEG** diagrams, dotted lines represent circles while solid lines represent straight lines, and all nodes, edges, and regions of the underlying graph are labeled for reference by numbers drawn in boxes.

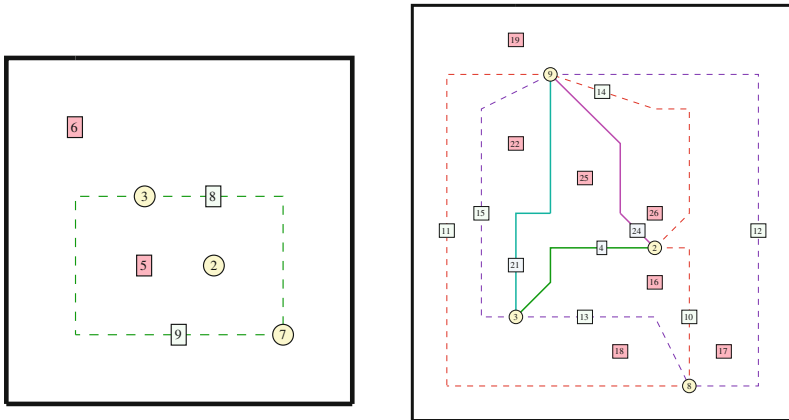


Fig. 1. Two CDEG diagrams.

Like more traditional formal systems that are sentential (that is, that operate on sentences that are strings of symbols in some formal language), **CDEG** has rules of inference that allow us to infer one diagram from another. In addition, and unlike traditional sentential formal systems, **CDEG** also has geometric construction rules. When one of these rules is applied to a diagram, we may get a set of several different possible resulting diagrams. **CDEG** uses a depth-first search algorithm to identify all the possible topologically distinct planar graphs that extend the starting graph by adding the newly constructed piece.

Once **CDEG** has determined the possible diagrams that result from one of its operations, it still has to lay them out in order to display them to the user. Each diagram is represented internally as a data structure that encapsulates its topological structure as a planar graph. When **CDEG** needs to display a diagram to the user, it relies on the open source library **OGDF**—the “Open Graph Drawing Framework” described in [2]—to lay out the diagrams using the Mixed Model algorithm of Gutwenger and Mutzel [3]. This algorithm works, but suffers from a number of disadvantages in this context. In particular, the diagrams would be much more readable if straight lines were represented by edges laid out in a straight line whenever possible, and if they didn’t change so much when new elements were added. Thus, one possible area for improvement of this system would be to identify a graph layout algorithm that resulted in diagrams that were easier for human users to interpret.

Another graph-theoretic challenge that arises in this context is “Lemma Incorporation”—how to automatically reuse previously proven results in new proofs. This is a trivial problem in traditional proof systems, but a difficult one in this diagrammatic setting, in which planar graphs have to be merged in an appropriate way. If diagram D_2 can be derived from D_1 in **CDEG**, we notate this by writing $D_1 \vdash D_2$. In order to use this result in a later proof, we would like to be able to apply it to any diagram D'_1 that contains D_1 as a subdiagram—that

is, from which D_1 can be obtained by erasing elements of the original diagram. We notate this by writing $D_1 \subset D'_1$. If $D_1 \vdash D_2$ and $D_1 \subset D'_1$, then we would like to be able to infer \mathcal{D}'_2 , where \mathcal{D}'_2 is the set of minimal diagrams that contain both D'_1 and D_2 as subdiagrams. A diagram D'_2 is minimal if it doesn't contain a subdiagram that still contains D'_1 and D_2 as subdiagrams. How to efficiently implement Lemma Incorporation into **CDEG** is an open question.

References

1. CDEG download page. <http://www.unco.edu/NHS/mathsci/facstaff/Miller/personal/CDEG/>
2. Chimani, M., Gutwenger, C., Jünger, M., Klein, K., Mutzel, P., Schulz, M.: The open graph drawing framework. In: 15th International Symposium on Graph Drawing 2007, GD 2007, Sydney, Australia (2007)
3. Gutwenger, C., Mutzel, P.: Planar polyline drawings with good angular resolution. In: Whitesides, S.H. (ed.) GD 1998. LNCS, vol. 1547, pp. 167–182. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-37623-2_13
4. Miller, N.: Euclid and His Twentieth Century Rivals: Diagrams in the Logic of Euclidean Geometry. CSLI Press, Stanford, CA (2007)

Drawing Graphs on Few Circles and Few Spheres

Myroslav Kryven¹(✉), Alexander Ravsky², and Alexander Wolff³ 

¹ Universität Würzburg, Würzburg, Germany
myroslav.kryven@uni-wuerzburg.de

² National Academy of Sciences of Ukraine, Lviv, Ukraine
alexander.ravsky@uni-wuerzburg.de

³ Universität Würzburg, Würzburg, Germany

A drawing of a given graph can be evaluated by many different quality measures depending on the concrete purpose of the drawing. Classical examples are the number of crossings, the ratio between the lengths of the shortest and the longest edge, or the angular resolution. Clearly, different layouts (and layout algorithms) optimize different measures. Hoffmann et al. [5] studied ratios between optimal values of quality measures implied by different graph drawing styles. They determined bounds for certain pairs of styles and showed that the ratio can be unbounded for others.

A few years ago, a new type of quality measure was introduced: the number of geometric objects that are needed to draw a graph given a certain style. Schulz [7] termed this measure the *visual complexity* of a drawing. More concretely, Dujmović et al. [4] defined the *segment number* of a graph G to be the minimum number of straight-line segments over all straight-line drawings of G . Similarly, Schulz [7] defined the *arc number* with respect to circular-arc drawings of G and showed that circular-arc drawings are an improvement over straight-line drawings not only in terms of visual complexity but also in terms of area consumption.

For our work, the most important precursor is the work of Chaplick et al. [2] who introduced another measure for the visual complexity of a graph G , namely the *plane cover number*, which is the minimum number of planes that together cover a straight-line drawing of G in three-dimensional space. Similarly, the *line cover number* of a planar graph G is the minimum number of lines that together cover a straight-line drawing of G in the plane. Among others, Chaplick et al. showed that the line cover number can be asymptotically smaller than the segment number, constructing n -vertex triangulations with line cover number $O(\sqrt{n})$ and segment number $\Theta(n)$.

Combining the approaches of Schulz and Chaplick et al., we define the *spherical cover number* of a graph G to be the minimum number of spheres such that G has a circular-arc drawing that is contained in the union of these spheres. Similarly, the *circular cover number* of a planar graph G is the minimum number of circles that together cover a circular-arc drawing of G on the sphere.

Any drawing with straight-line segments and circular arcs can be transformed into a circular-arc drawing by an inversion map. Therefore, we may

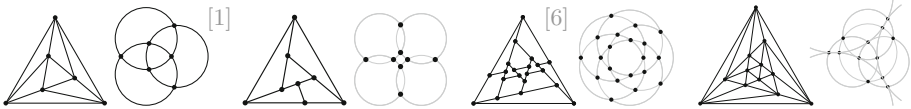
consider any line a “circle of infinite radius” and any plane a “sphere of infinite radius”. Hence, any affine cover can be considered a spherical cover.

We show that the sphere cover number of any n -vertex graph is $O(n)$, whereas Chaplick et al. [2] showed that the plane cover number of K_n is $\Theta(n^2)$.

Next, we analyze platonic graphs, that is, 1-skeletons of platonic solids. These graphs possess several nice properties: they are regular, planar and Hamiltonian. We use them as indicators to compare the above-mentioned measures of visual complexity. We have computed the following numbers (and two ranges):

$G = (V, E)$	$ V $	$ E $	$ F $	segment #	line cover #	arc # [ref.]	circular cover #
Tetrahedron	4	6	4	6	6	3	3
Cube	8	12	6	7	7	4	4
Octahedron	6	12	8	9	9	3	3
Dodecahedron	20	30	12	13	9 ... 10	10 [7]	5
Icosahedron	12	30	20	15	12 ... 15	7	7

For the upper bounds in the above table, we present drawings with optimal segment numbers, (near-) optimal line cover numbers, optimal arc cover numbers, and optimal circular cover numbers (we skip the tetrahedron):



For the lower bounds in the above table, we use various geometric and combinatorial arguments. For example, for the circle cover numbers, it is enough to count the minimum number of circles needed to accommodate the required number of vertices of given degrees. For the segment numbers, we set up an ILP that determines a locally consistent angle assignment [3] with the maximum number of π -angles between incident edges.

For all platonic graphs, we found *symmetric* circular-arc drawings with optimal circular cover and arc numbers, whereas it seems that the cube and the dodecahedron do not admit a symmetric straight-line drawing with optimal line cover or segment number. This is another advantage of (optimal) circular-arc drawings, apart from their smaller visual complexity.

References

1. Bekos, M.A., Raftopoulou, C.N.: Circle-representations of simple 4-regular planar graphs. In: Didimo, W., Patrignani, M. (eds.) GD 2012. LNCS, vol. 7704, pp. 138–149. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36763-2_13

2. Chaplick, S., Fleszar, K., Lipp, F., Ravsky, A., Verbitsky, O., Wolff, A.: Drawing graphs on few lines and few planes. In: Hu, Y., Nöllenburg, M. (eds.) GD 2016. LNCS, vol. 9801, pp. 166–180. Springer, Cham (2016). arxiv.org/abs/1607.01196
3. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall, Upper Saddle River (1999)
4. Dujmović, V., Eppstein, D., Suderman, M., Wood, D.R.: Drawings of planar graphs with few slopes and segments. *Comput. Geom. Theory Appl.* **38**, 194–212 (2007)
5. Hoffmann, M., van Kreveld, M., Kusters, V., Rote, G.: Quality ratios of measures for graph drawing styles. In: Proceedings of the 26th Canadian Conference on Computational Geometry, CCCG 2014, pp. 33–39 (2014)
6. Schemm, U.: Minimale Überdeckung von Knoten und Kanten in Graphen durch Geraden. Bachelor's Thesis, Institut für Informatik, Universität Würzburg (2016)
7. Schulz, A.: Drawing graphs with few arcs. *J. Graph Algorithms Appl.* **19**(1), 393–412 (2015)

Counterexample to the Variant of the Hanani–Tutte Theorem on the Genus-4 Surface

Radoslav Fulek¹(✉) and Jan Kynčl²

¹ IST Austria, Am Campus 1, 3400 Klosterneuburg, Austria
radoslav.fulek@gmail.com

² Department of Applied Mathematics and Institute
for Theoretical Computer Science, Faculty of Mathematics and Physics,
Charles University, Malostranské nám. 25, 118 00 Prague 1, Czech Republic
kync1@kam.mff.cuni.cz

The Hanani–Tutte theorem [5, 11] is a classical result that provides an algebraic characterization of planarity with interesting theoretical and algorithmic consequences, such as a simple polynomial algorithm for planarity testing [9]. The theorem has several variants, the strong and the weak variant are the two most well-known. The notion “the Hanani–Tutte theorem” refers to the strong variant.

Theorem (The (strong) Hanani–Tutte theorem [5, 11]). *A graph is planar if it can be drawn in the plane so that no pair of non-adjacent edges crosses an odd number of times.*

Theorem (The weak Hanani–Tutte theorem [1, 6, 8]). *If a graph G has a drawing \mathcal{D} on a compact surface \mathcal{S} where every pair of edges crosses an even number of times, then G has an embedding on \mathcal{S} that preserves the cyclic order of edges at each vertex of \mathcal{D} .*

Recently a common generalization of both the strong and the weak variant in the plane has been discovered.

Theorem (Unified Hanani–Tutte theorem [3, 8]). *Let G be a graph and let W be a subset of vertices of G . Let \mathcal{D} be a drawing of G where every pair of edges that are non-adjacent or have a common endpoint in W cross an even number of times. Then G has a planar embedding where cyclic orders of edges at vertices from W are the same as in \mathcal{D} .*

Pelsmayer, Schaefer and Stasi [7] extended the strong Hanani–Tutte theorem to the projective plane, using the list of forbidden minors. Colin de Verdière et al. [2] recently provided an alternative proof, which does not rely on the list of forbidden minors.

R. Fulek—Greatfully acknowledges support from Austrian Science Fund (FWF): M2281-N35.

J. Kynčl—Supported by project 16-01602Y of the Czech Science Foundation (GAČR).

Theorem (The Hanani–Tutte theorem on the projective plane [2, 7]). *If a graph G can be drawn on the projective plane so that no pair of non-adjacent edges crosses an odd number of times, then G can be embedded on the projective plane.*

It was an open problem if the strong Hanani–Tutte theorem extends to surfaces other than the plane and the projective plane. Furthermore, Schaefer and Štefankovič [10] conjectured that this is the case for all orientable surfaces.

Our results Our main result is a counterexample to the extension of the strong Hanani–Tutte theorem on the orientable surface of genus 4.

Theorem 1 *There exists a graph G that has a drawing in the compact orientable surfaces S with 4 handles in which every pair of non-adjacent edges cross an even number of times, but G cannot be embedded in S .*

Theorem 1 disproves a conjecture of Schaefer and Štefankovič [10, Conjecture 1] for \mathbb{Z}_2 -genus and genus, but the version for Euler \mathbb{Z}_2 -genus and Euler genus remains open. By taking a disjoint union of G from Theorem 1 with pairwise disjoint copies of K_5 we obtain a counterexample on an orientable surface of arbitrary genus bigger than 4.

In order to prove the theorem we first give a counterexample to the unified variant (see below) on the torus. Only part (1) of the following theorem is actually needed to prove Theorem 1, but (2) provides a good evidence for why the counterexample works.

Theorem 2

- (1) *The complete bipartite graph $K_{3,4}$ has a drawing \mathcal{D} on the torus with every pair of non-adjacent edges crossing an even number of times, such that for the set W of four vertices in one part every pair of edges with a common endpoint in W crosses an even number of times.*
- (2) *There is no embedding \mathcal{E} of $K_{3,4}$ on the torus with the same cyclic orders of edges at the vertices of W as in \mathcal{D} .*

The part (2) of Theorem 2 follows by an easy application of Euler’s formula once we observe that all the faces in the hypothetical embedding \mathcal{E} of $K_{3,4}$ must be of size at least 6.

Proof (of Theorem 1–sketch). The graph G is obtained by combining three disjoint copies of $K_{1,4}$ with a sufficiently large grid by appropriately identifying degree-1 vertices in the three copies of $K_{1,4}$ with vertices in the grid.

A drawing of G on the orientable surface of genus 4, in which every pair of non-adjacent edges cross an even number of times, is obtained as follows. We start by taking the toroidal drawing \mathcal{D} whose existence is claimed by Theorem 2, and drill 4 small holes around the vertices of W . The final drawing of G is obtained by gluing together the obtained torus with 4 holes containing the rest

of the drawing \mathcal{D} and an embedding of a large grid on a sphere with 4 holes along boundaries. The boundaries of the holes on the sphere are formed by 4-cycles.

In order to prove that G is not embeddable on \mathcal{S} we argue that by [4, Lemma 4] an embedding of G on \mathcal{S} must contain a large grid embedded in a planar way. This allows us to construct an embedding of $K_{4,5}$ on \mathcal{S} with minimal face of size 10 which cannot exist (contradiction). ■

References

1. Cairns, G., Nikolayevsky, Y.: Bounds for generalized thrackles. *Discrete Comput. Geom.* **23**(2), 191–206 (2000)
2. Colin de Verdière, É., Kaluža, V., Paták, P., Patáková, Z., Tancer, M.: A direct proof of the strong Hanani-Tutte theorem on the projective plane. In: Hu, Y., Nöllenburg, M. (eds.) *GD 2016*. LNCS, vol. 9801, pp. 454–467. Springer, Cham (2016)
3. Fulek, R., Kynčl, J., Pálvölgyi, D.: Unified Hanani-Tutte theorem. *Electron. J. Combin.* **24**(3)(P3.18), 8 (2017)
4. Geelen, J.F., Richter, R.B., Salazar, G.: Embedding grids in surfaces. *Eur. J. Combin.* **25**(6), 785–792 (2004)
5. Hanani, H.: Über wesentlich unplättbare Kurven im drei-dimensionalen Raume. *Fundamenta Mathematicae* **23**, 135–142 (1934)
6. Pach, J., Tóth, G.: Which crossing number is it anyway? *J. Combin. Theory Ser. B.* **80**(2), 225–246 (2000)
7. Pelsmayer, M.J., Schaefer, M., Stasi, D.: Strong Hanani-Tutte on the projective plane. *SIAM J. Discrete Math.* **23**(3), 1317–1323 (2009)
8. Pelsmayer, M.J., Schaefer, M., Štefankovič, D.: Removing even crossings. *J. Combin. Theory Ser. B.* **97**(4), 489–500 (2007)
9. Schaefer, M.: Geometry–intuitive, discrete, and convex, Hanani-Tutte and related results. In: János, P. (eds.) *Bolyai Society Mathematical Studies*, vol. 24, pp. 259–299. Budapest (2013)
10. Schaefer, M., Štefankovič, D.: Block additivity of \mathbb{Z}_2 -embeddings. In: Wismath, S., Wolff, A. (eds.) *GD 2013*. LNCS, vol. 8242, pp. 185–195. Springer, Cham (2013)
11. Tutte, W.T.: Toward a theory of crossing numbers. *J. Comb. Theor.* **8**, 45–53 (1970)

A Geometric Heuristic for Rectilinear Crossing Minimization

Marcel Radermacher¹(✉), Klara Reichard¹, Ignaz Rutter²,
and Dorothea Wagner¹

¹ Department of Computer Science, Karlsruhe Institute of Technology,
Karlsruhe, Germany

{radermacher,dorothea.wagner}@kit.edu

² Department of Mathematics and Computer Science, TU Eindhoven,
Eindhoven, The Netherlands
i.rutter@tue.nl

Introduction. The empirical study of Purchase [7] indicates that crossings have a major impact on the readability of drawings. Consequently, the minimization of crossings has received considerable attention in theory and in practice; the bibliography of Vrt'o is an impressive list of over 700 references [11].

In the case of topological drawings, where edges can be drawn as arbitrary Jordan arcs between their endpoints, iteratively inserting edges into a (planar) graph with a small number of crossings proved to be effective in practice [3]. However, this approach cannot be applied to straight-line drawings. Based on the topological drawings with a small number of crossings, Bläsius et al. [1] heuristically straighten the edges. Unfortunately, deciding whether there is a straight-line drawing homeomorphic to a given drawing is $\exists\mathbb{R}$ -complete [8]. Overall it is in general not possible to transfer the results on topological drawings to the geometric setting. Thus, if we insist on straight-line drawings, there is need for a geometric approach. For arbitrary graphs, we are only aware of one approach actively reducing crossings in straight-line drawings, the force-directed algorithm by Davidson and Harel [5].

Approach. Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E , and let Γ be a straight-line drawing of G . For a vertex $v \in V$ we denote by $\Gamma[v \mapsto p]$, $p \in \mathbb{R}^2$, the straight-line drawing obtained from Γ by moving v to the point p .

Theorem 1. *Let $G = (V, E)$ be a graph with $v \in V$ and a straight-line drawing Γ of G . A point $p^* \in \mathbb{R}^2$ such that $\text{cr}(\Gamma[v \mapsto p^*]) = \min_{q \in \mathbb{R}^2} \text{cr}(\Gamma[v \mapsto q])$ can be computed in $O((kn + m)^2 \log(kn + m))$ time with $k = \deg v$.*

Based on this primitive operation of moving a vertex to its crossing-minimal position, we introduce three heuristics in order to compute drawings with few

Work was partially supported by grant WA 654/21-1 of the German Research Foundation (DFG).

crossings. The *vertex movement approach* (VM) iteratively moves the vertices in a particular order to their locally optimal position. The *vertex insertion approach* (VI) starts from a large induced planar subgraph and inserts vertices at their locally optimal position. The *edge insertion approach* (EI and EP) starts with a maximal planar subgraph and iteratively inserts edges into the drawing and locally modifies the drawing to reduce the number of crossings. EP only moves the endpoints of the inserted edge, whereas EI also moves endpoints of edges that cross the inserted edge.

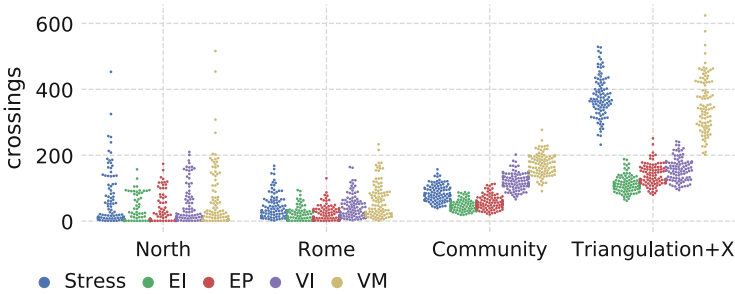


Fig. 1. Comparison of stress minimization and our heuristics.

Evaluation. We evaluated the energy-based algorithms implemented in OGDF [4] and our heuristics on four graph classes; (i) NORTH & ROME¹, (ii) COMMUNITY are graphs resembling community structure, and (iii) TRIANGULATION + X are maximal planar graphs with 64 vertices (generated using [2]) and ten additional random edges. The COMMUNITY graphs are generated with the LFR-GENERATOR [6] implemented in NETWORKKIT [9]. Our evaluation is based on hypotheses drawn from a scatter plot (Fig. 1) and are evaluated with a statistical test. The evaluation showed that stress minimization outperforms the remaining energy-based algorithms of OGDF, including the force-directed algorithm by Davidson and Harel [5]. The edge-insertion heuristics computes drawings with the smallest number of crossings, independent from the graph class. Especially, we observe that drawings obtained from energy-based algorithms applied on graphs in the class TRIANGULATION + X have a significantly larger number of crossings compared to graphs in the remaining classes. Stress minimization and the vertex-movement approach compute drawings with a similar number of crossings. Our statistical test shows that stress minimization computes drawings with about twice as many crossings as computed by our edge insertion approach. For graphs of the class TRIANGULATION+X the vertex-insertion approach computes drawings with half the crossings of stress in less than 30 s. Trading an even smaller number of crossings for a considerable increase of running time, edge insertion computes drawings with a third of the number of crossings compared

¹ <http://graphdrawing.org/data.html>.

to stress minimization on graphs of the class TRIANGULATION+X. The usage of precise geometric operations, provided by CGAL [10], has a negative influence on the running time of our heuristics. It is desirable to tune our implementation to handle larger instances.

References

1. Bläsius, T., Radermacher, M., Rutter, I.: How to draw a planarization. In: Steffen, B., Baier, C., van den Brand, M., Eder, J., Hinchey, M., Margaria, T. (eds.) SOFSEM 2017. LNCS, vol. 10139, pp. 295–308. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-51963-0_23
2. Brinkmann, G., McKay, B.D., et al.: Fast generation of planar graphs. *MATCH Commun. Math. Comput. Chem.* **58**(2), 323–357 (2007)
3. Buchheim, C., Chimani, M., Gutwenger, C., Jünger, M., Mutzel, P.: Handbook of graph drawing and visualization. In: *Crossings and Planarization*, pp. 43–85. Chapman and Hall/CRC (2013)
4. Chimani, M., Gutwenger, C., Jünger, M., Klau, G.W., Klein, K., Mutzel, P.: Handbook of graph drawing and visualization. In: *The Open Graph Drawing Framework (OGDF)*, pp. 543–569. Chapman and Hall/CRC (2013)
5. Davidson, R., Harel, D.: Drawing graphs nicely using simulated annealing. *ACM Trans. Graph.* **15**(4), 301–331 (1996)
6. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* **78**(4), 046110 (2008)
7. Purchase, H.C., Cohen, R.F., James, M.: Validating graph drawing aesthetics. In: Brandenburg, F.J. (ed.) *GD 1995*. LNCS, vol. 1027, pp. 435–446. Springer, Heidelberg (1996). <https://doi.org/10.1007/BFb0021827>
8. Schaefer, M.: Complexity of some geometric and topological problems. In: Eppstein, D., Gansner, E.R. (eds.) *GD 2009*. LNCS, vol. 5849, pp. 334–344. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11805-0_32
9. Staudt, C.L., Sazonovs, A., Meyerhenke, H.: Networkkit: An interactive tool suite for high-performance network analysis. arXiv preprint [arXiv:1403.3005](https://arxiv.org/abs/1403.3005) (2014)
10. The CGAL Project: CGAL User and Reference Manual. CGAL Editorial Board, 4.10 edn. (2017). <http://doc.cgal.org/4.10/Manual/packages.html>
11. Vrt’o, I.: Bibliography on crossing numbers of graphs (2014). <ftp://ftp.ifi.savba.sk/pub/imrich/crobib.pdf>

A Note on Plus-Contacts, Rectangular Duals, and Box-Orthogonal Drawings

Therese Biedl¹(✉) and Debajyoti Mondal²

¹ Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada
biedl@uwaterloo.ca

² Department of Computer Science, University of Saskatchewan, Saskatoon, Canada
dmondal@cs.usask.ca

A *plus-contact representation (PCR)* of an n -vertex planar graph G is a non-crossing arrangement of n plus shapes such that each vertex v of G is mapped to a distinct plus shape \oplus_v and two plus shapes touch if and only if the corresponding vertices are adjacent in G . If no arm of \oplus_v is incident to more than $c\Delta + O(1)$ other arms, where Δ is the maximum degree, then a PCR is called c -balanced; see Fig. 1(a)–(b). A *1-bend box-orthogonal drawing (BOD)* (resp., *1-bend Kandinsky drawing (KD)*) is a planar drawing where each vertex is drawn as an axis-aligned box (resp., square) and each edge is drawn as an orthogonal polyline with at most one bend between the corresponding boxes (resp., squares). Balanced PCRs can be transformed into 1-bend BODs or KDs [4], where vertices are drawn as squares of small side length; see Fig. 1(c).

Balanced plus-contact representations are motivated by the application of computing 1-bend BODs with boxes of small size and constant aspect ratio [8]. Besides, such representations have been useful to construct planar drawings with small number of distinct edge slopes [4, 6].

Contribution: In this poster we present the following result.

Theorem 1. *Every planar graph that admits a rectangular dual has the following representations: (a) A 1-bend BOD or KD, where each vertex v is a square of side length at most $(\deg(v)/2) + O(1)$. (b) A $(1/2)$ -balanced PCR, where for each vertex v , each arm of \oplus_v has at most $(\deg(v)/2) + O(1)$ contacts with other plus shapes.*

A graph admits a rectangular dual if and only if it is an irreducible triangulation (see e.g. [5]), i.e., a graph where the outer-face has degree at least 4, all inner faces are triangles, and there are no triangles that are not face.

The closest related results are 2-bend BODs where the length of the longer side of the box of v is at most $(\deg(v)/2) + O(1)$ [1], or 1-bend BODs, where the length of the longer side of the box of v is at most $\deg(v)$ [2]. Well balanced plus-contact representations are known only for 2-trees ($1/4 \leq c \leq 1/3$) and planar 3-trees ($1/3 \leq c \leq 1/2$) [4]. It is not known whether c -balanced PCRs

Work of the authors is supported in part by NSERC. See [3] for a full version.

exist for all planar graphs with $c < 1$, and it seems to be an interesting open question.

Computation of BOD: Given a rectangular dual \mathcal{R} , we first compute a *consistent rectangle contact representation* \mathcal{R}_c such that for every pair of adjacent rectangles $R_1, R_2 \in \mathcal{R}$, the corresponding $R_1, R_2 \in \mathcal{R}_c$ have the same (vertical or horizontal) adjacency and the line segment $R_1 \cap R_2$ lies entirely in one or both of R_1, R_2 . \mathcal{R}_c may contain four mutually adjacent rectangles and hence some unnecessary adjacencies. Let v be a vertex represented by rectangle $R \in \mathcal{R}_c$. We add inside R two polygonal zig-zag paths σ and σ' connecting the opposite corners of R ; see Fig. 1(d). Let the four *ords* of v be the four subpaths from the intersection point c to the corners of R . The crucial insight is that these ords (after a 45° -rotation) become axis-aligned zig-zag paths. Hence all bends can be removed using the topology-shape metric approach introduced by Tamassia [7]. Thus this shape is a plus shape \oplus_v with c at the center and the four ords becoming the four arms. We extend the ords of the neighbours of v to realize the required adjacencies, creating at most $(\deg(v)/2) + O(1)$ contacts on each cord of v . For example, among the rectangles incident to the top side of R , we can extend the bottom-left ords of half of them to touch the top-left cord of R , and the bottom-right ords of the remaining rectangles to touch the top-right cord of R . We call the resulting drawing a *pseudo-PCR*. The BOD is computed from this by first removing the bends using [7], then transforming the resulting PCR as explained in [4], and finally, by removing any unnecessary edge that may appear due to four mutually adjacent rectangles; see Fig. 1(e)–(h).

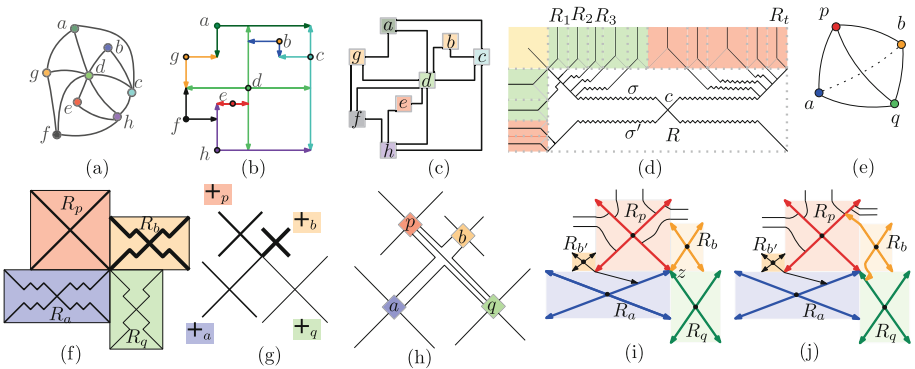


Fig. 1. (a)–(c) A $(1/2)$ -balanced PCR and a corresponding 1-bend BOD of a planar graph. (d) Construction of the cords. (e)–(h) Transformation into 1-bend BODs. (i)–(j) Modification of the pseudo-PCR. The plus shapes are drawn with bidirected lines. The thin lines represent the distribution of the incoming cords.

Balanced PCRs: To compute a $(1/2)$ -balanced PCR, we first compute the pseudo-PCR, and then remove the unnecessary adjacencies from it as follows. For every four mutually adjacent rectangles $R_a, R_p, R_b, R_q \in \mathcal{R}_c$, in this clockwise order around their common corner z , one of the edges (a, b) or (p, q)

does not belong to the input graph. We re-route the cords locally near R_p to remove the unnecessary adjacency. The details of processing R_p are unfortunately quite tedious; Fig. 1(i)–(j) show one of the many cases. Finally, we compute the required PCR by using the topology-shape metric approach [7].

References

1. Biedl, T., Kant, G.: A better heuristic for orthogonal graph drawings. *Comput. Geom.* **9**(3), 159–180 (1998)
2. Biedl, T.C., Kaufmann, M.: Area-efficient static and incremental graph drawings. In: Burkard, R., Woeginger, G. (eds.) *ESA 1997*. LNCS, vol. 1284, pp. 37–52. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63397-9_4
3. Biedl, T., Mondal, D.: A note on plus-contacts, rectangular duals, and box-orthogonal drawings. *CoRR* abs/1708.09560 (2017). <https://arxiv.org/abs/1708.09560>
4. Durocher, S., Mondal, D.: On balanced -contact representations. In: Wismath, S., Wolff, A. (eds.) *GD 2013*. LNCS, vol. 8242, pp. 143–154. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03841-4_13
5. Fusy, É.: Transversal structures on triangulations: a combinatorial study and straight-line drawings. *Discrete Math.* **309**(7), 1870–1894 (2009)
6. Di Giacomo, E., Liotta, G., Montecchiani, F.: 1-Bend upward planar drawings of SP-digraphs. In: Hu, Y., Nöllenburg, M. (eds.) *GD 2016*. LNCS, vol. 9801, pp. 123–130. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_10
7. Tamassia, R.: On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.* **16**(3), 421–444 (1987)
8. Wood, D.R.: Multi-dimensional orthogonal graph drawing with small boxes. In: Kratochvíl, J. (ed.) *GD 1999*. LNCS, vol. 1731, pp. 311–322. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-46648-7_32

Grid Obstacle Representation of Graphs

Arijit Bishnu¹, Arijit Ghosh¹, Rogers Mathew², Gopinath Mishra¹,
and Subhabrata Paul³(✉)

¹ Indian Statistical Institute, Kolkata, India

² Indian Institute of Technology, Kharagpur, India

³ Indian Institute of Technology, Patna, India
paulsubhabrata@gmail.com

1 Introduction

In 2010, Alpert et al. [1] introduced the concept of *obstacle representation* of a graph G . The obstacle representation of G is about assigning points in \mathbb{R}^2 for each vertex of G and blocking the visibility among pairs of points whose corresponding vertices do not have an edge. In the Euclidean plane, the shortest path and straight line visibility are essentially the same. We introduce a new definition of obstacle representation in \mathbb{Z}^d as follows; this can be generalized to any metric space as given in [3, 4].

Definition 1 *The grid obstacle representation of a graph $G = (V, E)$ is an injective map $f : V \rightarrow \mathbb{Z}^d$ and a set of point obstacles \mathcal{O} on grid points of $\mathbb{Z}^d \setminus f(V)$ such that uv is an edge in G if and only if there exists a Manhattan path between $f(u)$ and $f(v)$ in \mathbb{Z}^d avoiding the obstacles of \mathcal{O} . The grid obstacle number of a graph is the smallest number of obstacles needed for the grid obstacle representation of G .*

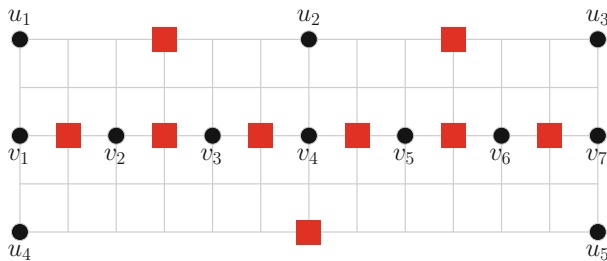


Fig. 1. Grid obstacle representation of $K_{n,m}$; the size of the grid and the number of obstacles is $O(n + m)$. The dots represent the vertices and the squares represent the obstacles.

2 Existence and Non-existence Results

We use algorithms for straight line embeddings of planar graphs in \mathbb{Z}^2 [5, 9] and of arbitrary graphs in \mathbb{Z}^3 [8] to prove the following results.

Theorem 1 ([3, 4])

1. Every planar graph with n vertices admits a \mathbb{Z}^2 grid obstacle representation in a $O(n^4) \times O(n^4)$ grid.
2. Every r colorable graph with n vertices admits a \mathbb{Z}^3 grid obstacle representation in a $O(r^4 n^3) \times O(r^3 n^4) \times O(r^4 n^4)$ grid.

Biedl and Mehrabi [2] in a follow-up work improved the size of the grid required for a grid obstacle representation in \mathbb{Z}^2 and \mathbb{Z}^3 .

We also study the grid obstacle representation of a graph G in a *horizontal strip*. A *horizontal strip* is a grid where the y -coordinates are bounded but the x -coordinates can be arbitrary integer. We can show another existential result that says that if a graph has a grid obstacle representation in a horizontal strip, then it can be embedded inside a bounded grid using a *compression technique*.

Theorem 2 ([4]) *Let G admit a grid obstacle representation in a horizontal strip of height b . Then G has a grid obstacle representation in a $b \times O(b^3 n)$ grid.*

Pach [7] resolved a question raised in an earlier manuscript of ours [4] by showing that there exists bipartite graphs with no grid obstacle representation in \mathbb{Z}^2 . We proved the following non-existence result.

Theorem 3 ([4]) *There exists a non-quasiplanar C_4 -free graph on more than 20 vertices (having at least $8n - 19$ edges, where n denotes the number of vertices in the graph G) which does not admit a grid obstacle representation in \mathbb{Z}^2 .*

3 Hardness Results

Here, we study a problem of ℓ_1 -obstacle representability on a given point set (ℓ_1 -OEPS) of a graph. The input instance is a graph $G = (V, E)$ and a set S , of size $|V|$, that is a subset of a grid whose size is polynomial in $|V|$. The problem is to decide whether there exists an ℓ_1 -obstacle representation of G such that the vertices of G are mapped to S . Now, we show that ℓ_1 -OEPS is NP-complete for subdivision of non-Hamiltonian planar cubic graphs. The reduction is from a restricted version of *geodesic point set embeddability* problem. The problem is whether a planar graph has a Manhattan-geodesic drawing such that the vertices are embedded onto a given set of points S . In the restricted version of geodesic point set embeddability problem ((P_0, P_1, P_2) -GPSE), the given point set S is partitioned into three sets, P_0, P_1 and P_2 , where $P_0 = \{(-j, 0) | j = 0, 1, \dots, 2n - 2\}$, $P_1 = \{(j, nj) | j = 1, 2, \dots, k_1\}$, and $P_2 = \{(j, -nj) | j = 1, 2, \dots, k_2\}$ with $k_1 + k_2 = n/2 + 1$. This restricted version is known to be NP-complete [6] for subdivision of non-Hamiltonian planar cubic graphs.

Theorem 4 ([4]) *ℓ_1 -OEPS is NP-complete for subdivision of planar cubic graphs.*

References

1. Alpert, H., Koch, C., Laison, J.D.: Obstacle numbers of graphs. *Discrete Comput. Geom.* **44**(1), 223–244 (2010)
2. Biedl, T., Mehrabi, S.: Grid-obstacle representations with connections to staircase guarding (2017). ArXiv e-prints, abs/1708.01903
3. Bishnu, A., Ghosh, A., Mathew, R., Mishra, G., Paul, S.: Grid obstacle representation of graphs. Manuscript (2015)
4. Bishnu, A., Ghosh, A., Mathew, R., Mishra, G., Paul, S.: Grid obstacle representation of graphs (2017). ArXiv e-prints, abs/1708.01765
5. Fraysseix, H.D., Pach, J., Pollack, R.: How to draw a planar graph on a grid. *Combinatorica* **10**(1), 41–51 (1990)
6. Katz, B., Krug, M., Rutter, I., Wolff, A.: Manhattan-geodesic embedding of planar graphs. In: Eppstein, D., Gansner, E.R. (eds.) GD 2009. LNCS, vol. 5849, pp. 207–218. Springer, Heidelberg (2009)
7. Pach, J.: Graphs with no grid obstacle representation. *Geombinatorics* **26**(2), 80–83 (2016)
8. Pach, J., Thiele, T., Tóth, G.: Three-dimensional grid drawings of graphs. In: DiBattista, G. (eds.) GD 1997. LNCS, vol. 1353, pp. 47–51. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63938-1_49
9. Schnyder, W.: Embedding planar graphs on the grid. In: Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 138–148 (1990)

Summarizing and Visualizing Graph Ensembles with Rank Statistics and Boxplots

Mukund Raj¹(✉), Ian Ruginski², Robert M. Kirby¹, and Ross T. Whitaker¹

¹ School of Computing, University of Utah, Salt Lake City, USA

² Department of Psychology, University of Utah, Salt Lake City, USA

1 Introduction

The problem of visualizing graphs becomes more complex as we consider the growing diversity of visualization tasks on graphs. For instance, in the domain of neuroscience, there is a need to gain insight into how a graph (representing a brain network) compares to another, or how a graph ensemble (brain networks associated with a specific group) compares to an individual graph or another ensemble [1]. The goal of this project is to develop a method to visualize graph ensembles in a way that is able to convey the underlying structure (both the center and variability of the underlying distribution of edge weights) in context of the relationships between nodes in the graph. Specifically, we hope to help accomplish two important tasks that pertain to applications involving weighted graph ensembles: (1) comparing two different graph ensembles and (2) comparing individual members relative to an ensemble. We limit the scope of this project to graphs ensembles that share common vertex/edge sets and differ only with regard to edge weights.

2 Method

We propose a visualization method, called *network boxplot*, for visualizing graph ensembles. The first step for drawing a network boxplot, analogous to the traditional Tukey boxplot and other existing methods for summarizing ensembles [3–5], is to compute center outward order and rank statistics for the members in the ensemble. We use a discrete functional representation of graph adjacency matrices which allows us to use the functional band depth (see [2]) to obtain required statistics for members of the ensemble. The second step is to generate a visualization using the order and rank statistics. Figure 1a shows a rendering of the network boxplot. We employ an adjacency matrix representation, and use cells in a *single* matrix to display the summary statistics for the ensemble. Each cell in a network boxplot encodes the weight on the median graph and the extent of weights on graphs in the 50% band for corresponding edges in the graph ensemble. The weight on median graph is encoded in two ways: the background color of the cell as well as the radius of circle between the two gray rings (annuli). The upper and lower extents of the 50% bands are encoded by the outer and inner gray rings.

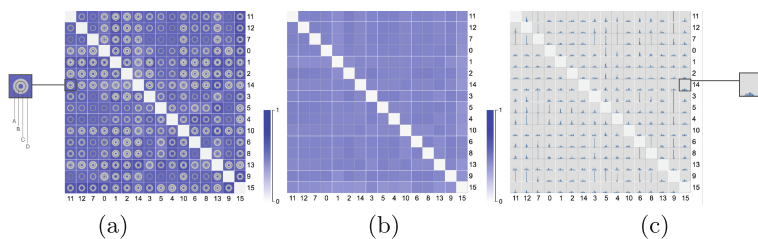


Fig. 1. Visualizing graph ensembles. (a) A network boxplot visualization. ‘A’ and ‘B’ (anuli) indicate the 50% band while ‘C’ (circle radius) and ‘D’ (background color) are two different encodings of the median. (b) Heatmap, and (c) cell histogram for weighted adjacency matrix ensemble

3 Results and Discussion

We conducted a pilot user study to evaluate the performance of the proposed network boxplot visualization in the task of comparing two graph ensembles (edge weights were generated using Gaussian processes). We found that participants made more accurate—although slower—judgments using the proposed method relative to the existing methods—namely, heatmap (Fig. 1b) and cell histogram [6] (Fig. 1c). A key advantage of our approach over existing methods is the ability to convey correlations across edges. We plan to conduct a larger user study which would also include the task of comparing an individual graph to an ensemble. We have also developed a network boxplot based interactive system to explore real brain fMRI network ensemble data. Our plan is to work with domain experts to evaluate the system and improve its effectiveness.

Acknowledgments. This work was supported by National Science Foundation (NSF) grant IIS-1212806.

References

1. Alper, B., Bach, B., Henry Riche, N., Isenberg, T., Fekete, J.D.: Weighted graph comparison techniques for brain connectivity analysis. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 483–492. ACM (2013)
2. López-Pintado, S., Romo, J.: On the concept of depth for functional data. *J. Am. Stat. Assoc.* **104**(486), 718–734 (2009)
3. Mirzargar, M., Whitaker, R., Kirby, R.: Curve boxplot: generalization of boxplot for ensembles of curves. *IEEE Trans. Vis. Comput. Graph.* **20**(12), 2654–2663 (2014)
4. Sun, Y., Genton, M.G.: Functional boxplots. *J. Comput. Graph. Stat.* **20**(2), 316–334 (2011)

5. Whitaker, R.T., Mirzargar, M., Kirby, R.M.: Contour boxplots: a method for characterizing uncertainty in feature sets from simulation ensembles. *IEEE Trans. Vis. Comput. Graph.* **19**(12), 2713–2722 (2013)
6. Yi, J.S., Elmqvist, N., Lee, S.: Timematrix: analyzing temporal social networks using interactive matrix-based visualizations. *Int. J. Human-Comput. Interact.* **26**(11–12), 1031–1051 (2010)

Planar k -NodeTrix Graphs

A New Family of Beyond Planar Graphs

Emilio Di Giacomo¹, Giuseppe Liotta¹, Maurizio Patrignani²,
and Alessandra Tappini¹(✉)

¹ Università degli Studi di Perugia, Perugia, Italy
{emilio.digiacomo,giuseppe.liotta}@unipg.it,
alessandra.tappini@studenti.unipg.it

² Roma Tre University, Rome, Italy
patrigna@dia.uniroma3.it

Introduction. Motivated by the problem of visualizing non-planar graphs, the so-called *beyond planarity* has become one of the most studied graph-drawing topics in the last years. Several families of beyond-planar graphs have been defined by imposing restrictions on the number or type of edge crossings (see, e.g., [2, 5, 6, 8, 9]). Another emerging graph drawing paradigm for the visualization of non-planar graphs is *hybrid planarity* [3, 7]. In a hybrid planar drawing dense subgraphs (*clusters*), for which a node-link representation would not be effective, are visualized with an alternative type of representation, and these clusters are connected with edges that do not cross each other. Inspired by the NodeTrix representations proposed by Henry et al. [7], *planar NodeTrix representations* have been studied. A planar NodeTrix representation is a hybrid planar drawing where clusters are represented by adjacency matrices. Batagelj et al. [1] studied the problem of computing the clusters so that the NodeTrix representation is planar, while Da Lozzo et al. [3] investigated the problem of testing a graph for NodeTrix planarity (see also [4]). In this poster we consider NodeTrix planarity from another perspective. We study the properties of *planar k -NodeTrix graphs*, i.e., graphs that admit a planar NodeTrix representation where the matrices have size at most k . We also define and study a new graph parameter, the *planar NodeTrix number*, which is the minimum k for which a graph is planar k -NodeTrix.

Main Results. In this section, we give a detailed list of our main results. First of all, we prove a tight bound on the density of planar k -NodeTrix graphs.

Theorem 1. *A graph with n vertices and m edges admits a planar k -NodeTrix representation only if $m \leq n(\frac{k}{2} + \frac{7}{2} + \frac{1}{k}) - 6$. Furthermore, for every pair of integers $i \geq 3$ and $k \geq 2$, there exists a planar k -NodeTrix graph $\overline{G}_{i,k}$ with $n = 3k \cdot i$ vertices and $n(\frac{k}{2} + \frac{7}{2} + \frac{1}{k}) - 6$ edges.*

We then study the relationship between planar k -NodeTrix graphs and other families of non-planar graphs. We show that the families of planar 2-NodeTrix graphs and 1-planar graphs have a non-empty intersection, but no one is contained into the other. In a 1-planar graph each edge is crossed by at most

one other edge. Thus, it seems reasonable to remove each crossing by merging, in a matrix of size 2, two of the vertices that are involved in the crossing. The next two theorems show that in general this is not the case. An *optimal 1-planar graph* is a 1-planar graph with exactly $4n - 8$ edges, which is the maximum number of edges that an n -vertex 1-planar graph can have. A *NIC-planar graph* (resp. *IC-planar graph*) is a 1-planar graph that admits a drawing such that every two pairs of crossing edges share at most one vertex (resp. share no vertex).

Theorem 2. *There exists an optimal 1-planar graph G with $n = 28$ vertices and $m = 104$ edges such that $\text{nt}(G) > 2$.*

Theorem 3. *For every $h > 2$, there exists a NIC-planar graph H_h with $n = 5 \cdot 2^h - 8$ vertices and $m = 18 \cdot 2^h - 36$ edges such that $\text{nt}(H_h) > 2$.*

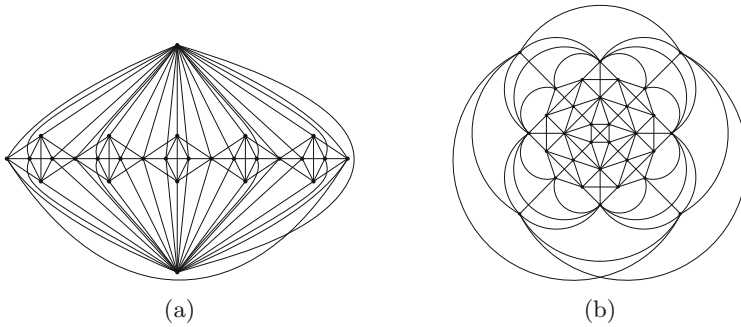


Fig. 1. (a) An optimal 1-planar graph G with $\text{nt}(G) > 2$. (b) A NIC-planar graph H_3 with $\text{nt}(H_3) > 2$.

Figure 1 shows an optimal 1-planar graph and a NIC-planar graph with planar NodeTrix number larger than 2.

In contrast with the previous theorems, the next result shows a family of NIC-planar graphs that has planar NodeTrix number 2. Let G be a NIC-plane graph and let $\langle e_1, f_1 \rangle, \dots, \langle e_h, f_h \rangle$ be the pairs of crossing edges of G . The *crossing pairs graph* (shortened as *cp-graph*) of G is a graph with a vertex w_i for each pair $\langle e_i, f_i \rangle$ and an edge between w_i and w_j if $\langle e_i, f_i \rangle$ and $\langle e_j, f_j \rangle$ share a vertex. An *AcNIC-planar graph* is a NIC-planar graph whose *cp-graph* is acyclic. Notice that AcNIC-planar graphs include the IC-planar graphs because the *cp-graph* of an IC-planar graph has no edge.

Theorem 4. *Every AcNIC-planar graph has planar NodeTrix number two.*

Finally, we show that the planar NodeTrix number of K_n is at most $n - 4$ and that this bound is tight for $n \geq 32$.

Theorem 5. *For $n > 5$, $\text{nt}(K_n) \leq n - 4$ and for $n \geq 32$, $\text{nt}(K_n) = n - 4$.*

References

1. Batagelj, V., Brandenburg, F., Didimo, W., Liotta, G., Palladino, P., Patrignani, M.: Visual analysis of large graphs using (x, y) -clustering and hybrid visualizations. *IEEE Trans. Vis. Comput. Graph.* **17**(11), 1587–1598 (2011)
2. Binucci, C., Di Giacomo, E., Didimo, W., Montecchiani, F., Patrignani, M., Symvonis, A., Tollis, I.G.: Fan-planarity: properties and complexity. *TCS* **589**, 76–86 (2015)
3. Da Lozzo, G., Di Battista, G., Frati, F., Patrignani, M.: Computing NodeTrix representations of clustered graphs. In: Hu, Y., Nöllenburg, M. (eds.) *GD 2016*. LNCS, vol. 9801, pp. 107–120. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_9
4. Di Giacomo, E., Liotta, G., Patrignani, M., Tappini, A.: Planar k -NodeTrix graphs. In: Frati, F., Ma, K.-L. (eds.) *GD 2017*. LNCS, vol. 10692, pp. 479–491. Springer, Cham (2017)
5. Didimo, W., Eades, P., Liotta, G.: Drawing graphs with right angle crossings. *Theoret. Comput. Sci.* **412**(39), 5156–5166 (2011)
6. Fox, J., Pach, J., Suk, A.: The number of edges in k -quasi-planar graphs. *SIAM J. Discrete Math.* **27**(1), 550–561 (2013)
7. Henry, N., Fekete, J., McGuffin, M.J.: NodeTrix: a hybrid visualization of social networks. *IEEE Trans. Vis. Comput. Graph.* **13**(6), 1302–1309 (2007)
8. Kobourov, S.G., Liotta, G., Montecchiani, F.: An annotated bibliography on 1-planarity. *CoRR* abs/1703.02261 (2017)
9. Pach, J., Tóth, G.: Graphs drawn with few crossings per edge. *Combinatorica* **17**(3), 427–439 (1997)

Towards Characterizing Strict Outerconfluent Graphs

Fabian Klute^(✉) and Martin Nöllenburg

Algorithms and Complexity Group, TU Wien, Vienna, Austria

Confluent drawings of graphs are geometric representations in the plane, in which vertices are mapped to points, but edges are not drawn as individually distinguishable geometric objects. Instead, an edge is represented by the presence of a smooth curve between two vertices in a system of arcs and junctions.

More formally, a confluent drawing D of a graph $G = (V, E)$ consists of a set of points representing the vertices, a set of junction points, and a set of smooth arcs, such that each arc starts and ends at a vertex point or a junction, no two arcs intersect (except at common endpoints), and all arcs meeting in a junction share the same tangent line in the junction point. There is an edge $(u, v) \in E$ if and only if there is a smooth path from u to v in D that does not pass through any other vertex.

Confluent drawings were introduced by Dickerson et al. [1], who identified classes of graphs that admit or that do not admit confluent drawings. Later, variations such as strong and tree confluency [6], as well as Δ -confluency [2] were introduced. Confluent drawings have further been used for layered drawings [3] and for drawing Hasse diagrams [5]. The complexity of the recognition problem for graphs that admit a confluent drawing remains open.

Eppstein et al. [4] defined *strict* confluent drawings, in which every edge of the graph must be represented by a unique smooth path. They showed that for general graphs it is NP-complete to decide whether a strict confluent drawing exists. A strict confluent drawing is called *strict outerconfluent* if all vertices lie on the boundary of a (topological) disk that contains the strict confluent drawing. For a given cyclic vertex order, Eppstein et al. [4] presented a constructive poly-time algorithm for testing the existence of a strict outerconfluent drawing. Without a given vertex order the recognition complexity as well as a characterization of the graphs admitting such drawings remained open. We present first results towards characterizing the strict outerconfluent (SOC) graphs by examining potential sub- and super-classes of SOC graphs. For definitions of the used graph classes we refer to www.graphclasses.org.

If we draw a graph G as a traditional circular drawing with straight-line edges, then all the crossings are determined by the order of the vertices alone. We can replace a crossing by a confluent junction if the two edges forming the crossing are part of a $K_{2,2}$. We call such a crossing *represented*. It is clear that a graph can only have a strict outerconfluent drawing if it has a circular layout with all crossings represented. This is not sufficient though, as there are such graphs that

have no strict outerconfluent drawing. We obtain two 6-vertex obstructions for strict outerconfluent drawings, namely a $K_{3,3}$ with an alternating vertex order and a domino graph (two four-cycles sharing an edge) in bipartite order.

Our next result concerns bipartite drawings. Let $G = (X, Y, E)$ be a bipartite graph with vertex sets X and Y . We call a strict outerconfluent drawing D a *bipartite strict outerconfluent drawing* if the nodes can be partitioned into two independent sets, such that each set is consecutive on the boundary of the topological disk. Hui et al. [6] showed that the bipartite outerconfluent graphs are exactly the bipartite permutation graphs. We show that the (bipartite-permutation \cap domino-free) graphs are exactly the bipartite strict outerconfluent graphs. The proof uses the drawing algorithm by Hui et al. to obtain a confluent bipartite drawing, which is non-strict if and only if a domino is present.

On the other hand we show that circle and comparability graphs are neither sub- nor superclasses of the SOC graphs and the alternation and circle-polygon graphs are no sub-classes of them. All the results can be shown via counterexamples, mostly using the wheel on six vertices and the so-called BW_3 graph, which both have no SOC drawing.

Finally our main result shows an interesting superclass of SOC graphs. The class of *outer-string* graphs contains all graphs $G = (V, E)$ which can be represented by an intersection model of curves in a disk with one end-point on the disk's boundary. We show that SOC graphs are outer-string graphs. The inclusion is proper, because not every circle-polygon graph is an SOC graph, but every circle-polygon graph is an outer-string graph.

Let D be a strict outerconfluent drawing. To get an outer-string representation of the corresponding graph G_D we need to find for every vertex v in G_D a string starting at the node representing v in D and intersecting only strings representing adjacent vertices in G_D . We do this by exploiting the tree structure we get for one node in D , when looking at all the junctions and other nodes which can be reached from it via smooth paths. We call a junction j *split-junction*, if the path coming from v separates at j into two paths and *merge-junction* if another path fuses with it at j . One string is then constructed as follows:

- Start from a node and traverse its tree in left-first DFS order
- At leaf, make a clockwise U-turn and backtrack to the previous split-junction.
- At split-junction:
 - coming from the left subtree: cross the arc from the left subtree at the junction and descend into the right subtree
 - coming from the right subtree: cross the arc to the left subtree and backtrack along the existing string to the previous split-junction

To find the complete outer-string representation of G_D we have to combine all these strings for nodes in D . We distinguish three cases, two of which are straightforward. If two nodes are connected by a path we have to guarantee that the two strings intersect at least once, which can be done at the leaves. The second one considers two nodes without a path connecting them and the

two trees are independent, i.e., not sharing a junction. Then the strings are independent by construction as well. Finally if the trees share junctions, then these can be only merge-junctions. The key observation here is that at most two merge-junctions can be shared by two nodes without a connecting path in D .

References

1. Dickerson, M., Eppstein, D., Goodrich, M.T., Meng, J.Y.: Confluent drawings: visualizing non-planar diagrams in a planar way. *J. Graph Algorithms Appl.* **9**(1), 31–52 (2005)
2. Eppstein, D., Goodrich, M.T., Meng, J.Y.: Delta-confluent drawings. In: Healy, P., Nikolov, N.S. (eds.) *GD 2005*. LNCS, vol. 3843, pp. 165–176. Springer, Heidelberg (2006). https://doi.org/10.1007/11618058_16
3. Eppstein, D., Goodrich, M.T., Meng, J.Y.: Confluent layered drawings. *Algorithmica* **47**, 439–452 (2007)
4. Eppstein, D., Holten, D., Löffler, M., Nöllenburg, M., Speckmann, B., Verbeek, K.: Strict confluent drawing. *J. Comput. Geom.* **7**(1), 22–46 (2016)
5. Eppstein, D., Simons, J.A.: Confluent hasse diagrams. *J. Graph Algorithms Appl.* **17**(7), 689–710 (2013)
6. Hui, P., Pelsmajer, M.J., Schaefer, M., Stefankovic, D.: Train tracks and confluent drawings. *Algorithmica* **47**(4), 465–479 (2007)

Flattening Polygonal Linkages via Uniform Angular Motion

Hugo A. Akitaya¹, Matthew D. Jones¹, Gregory A. Sandoval²,
Diane L. Souvaine¹, David Stalfa¹, and Csaba D. Tóth^{1,2}(✉)

¹ Tufts University, Medford, MA, USA

{hugo.alves_akitaya,matthew.jones,diane.souvaine,david.stalfa}@tufts.edu

² California State University Northridge, Los Angeles, CA, USA

{gregory.sandoval.3,csaba.toth}@csun.edu

Abstract. We study the motion of polygonal linkages under the restriction that the angles between adjacent edges change uniformly to 0 , π , or 2π . We show that convex polygons, orthogonally convex polygons, and orthogonal 2-terrains unfold without self-intersection to a straight line in this model, but there exists an orthogonal 12-gon that does not. Further, we show that regular polygons, triangles, quadrilaterals, and convex pentagons can be reconfigured into flat zigzag chains; and every $m \times n$ rectangle made of unit-length edges can be reconfigured into a unit-length zigzag.

1 Introduction

A polygonal linkage is a graph embedded in the plane where the edges are *rigid bars* and the vertices are *joints* between adjacent edges. By the classical Carpenter’s Rule Theorem, every crossing-free path can be reconfigured continuously into a straight-line segment, and every simple polygon into a convex polygon. However, there are configurations that are locked, in the sense that the configuration space is disconnected [1–3]. In some applications, the reconfiguration of a linkage is controlled by physical parameters (e.g., change in temperature), and these parameters equally impact all joints of the linkage. This motivates the study of the following model. Consider a crossing-free linkage, where the *angle* between every pair of adjacent edges is a *linear* function of time.

We explore the configuration space of linkages in this model and obtain feasibility and infeasibility results. Ultimately, we would like to characterize the “shapes” (defined as the outer face of a graph) that can be obtained from a “flat” linkage (i.e., a polygonal linkage in which all edges are collinear) without self-intersection. Given a simple polygon P , we fit a polygonal linkage on P such that its two endpoints coincide with v_0 . We can choose vertex v_0 and target values 0 , π , or 2π for the interior angles of the linkage. Since the angles change uniformly, these parameters determine the motion of the linkage up to congruences. We wish to find parameters that yield a motion without self-intersection.

Research partially supported by the NSF awards CCF-1422311 and CCF-1423615, and the Science Without Borders scholarship program.

2 Unfolding Polygonal Linkages into a Straight Line

Assume that the flat angle π is the target value for all angles of the linkage. We show that for a convex polygon, an orthogonally convex polygon, and an orthogonal 2-terrain, one can choose a vertex v_0 such that the boundary of the polygon unfolds without self-intersection to a straight line in this model.

We partition an orthogonal convex polygon into four staircases and show that they each unfold without self-intersection. The strategy is to open the polygon into a path at one of the two highest vertices. By case analysis we show that the different staircases do not intersect each other. For any two edges in opposite staircases we find a separating line and show that they remain in opposite halfplanes. Our result extends to orthogonal polygons composed of up to six staircases. However, there are orthogonal polygons composed of eight staircases that would self-intersect no matter at which vertex we open it into a path.

We define an orthogonal 2-terrain as an x -monotone orthogonal polygon such that there is a horizontal internal chord that connects the leftmost and rightmost edges. We open the polygon at one of the leftmost vertices and show that both the upper and lower chains remain monotone and lie in disjoint halfplanes throughout the motion. Opening a convex polygon at an arbitrary vertex defines an expansive motion and therefore the linkage would unfold without self-intersection.

3 Reconfiguring Polygonal Linkages into a Zigzag

For reconfiguring a polygonal linkage into zigzag, the main strategy is to partition the linkage into subchains, and allocate disjoint regions to the subchains, then show that the subchains remain in their own regions, and that none of the subchains self-intersect. For example, consider the simplest case, where the linkage forms a triangle. Let abc be a triangle such that ab lies on the positive x -axis, b is at the origin, and c is above the x -axis. Assume we open abc into a polygonal chain a_1bca_2 , where $\angle a_1bc$ goes to 0 and $\angle bca_2$ to 2π . If a_1b is fixed, we can show that the y -coordinate of a_2 remains nonnegative, hence a_2 remains above the x -axis at all times. The same strategy works for quadrilaterals by treating each of the two pairs of edges in the quadrilaterals as sides of triangles formed with a diagonal. Convex pentagons can be handled in the same fashion, where a diagonal divides the pentagon into a convex quadrilateral and a triangle.

The case of regular polygons is slightly more complicated. The rotational symmetry makes it most effective to use pairs of consecutive edges as subchains and partition the plane radially from a point at the common intersection of angle bisectors at all times. For m -by- n rectangles formed by unit segments, we handle different cases based on the parities of m and n . The simplest case is when n is even, where the plane can be partitioned using two vertical rays. The most complicated case where m and n are both odd is handled using a more complicated partition.

If we can choose each angle to go to 0, π , or 2π , the computational complexity of deciding whether a given linkage self-intersects remains open.

References

1. Connelly, R., Demaine, E.D.: Geometry and topology of polygonal linkages. In: Handbook of Discrete and Computational Geometry, 3rd edn., pp. 233–256. CRC Press, Boca Raton (2017)
2. Demaine, E.D., O’Rourke, J.: Geometric Folding Algorithms: Linkages, Origami, and Polyhedra. Cambridge University Press, Cambridge (2007)
3. O’Rourke, J.: How to Fold It: The Mathematics of Linkages, Origami, and Polyhedra. Cambridge University Press, Cambridge (2011)

Optimal Compaction of Orthogonal Grid Drawings for Graphs of Arbitrary Vertex Degrees

Eduardo Santiago Ramos^{1,2}(✉) and Adriano Chaves Lisboa^{1,2}

¹ Gaia, solutions on demand, Belo Horizonte, Brazil
{eduardo.ramos,adriano.lisboa}@gaiasd.com

² Universidade Federal de Minas Gerais, Belo Horizonte, Brazil
<http://www.ufmg.br>
<http://www.gaiasd.com>

Abstract. Orthogonal graphs are used in a multitude of applications to visualize information. Examples include database design, software engineering, VLSI layout and UML diagrams. The TSM approach is an effective methodology for creating orthogonal grid drawings of graphs. Its name is an acronym of its three stages: *topology*, in which a planar representation is defined; *shape*, when an orthogonal representation is obtained; and *metrics*, in which the graph's elements are positioned on the grid in accordance to the orthogonal representation, while optimizing some characteristic of the drawing.

Regarding the *metrics* stage, in 1998, Klau and Mutzel [5] presented an integer linear programming formulation for the problem that performs two-dimensional compaction and yields optimum results. This was a major accomplishment, not only for its optimality (given that the compaction problem was proven to be NP-complete [8]), but also for its description of characteristics that apply to any correct layout. Additionally, it presents a great deal of flexibility [1] and extensibility [3]. The quality of this method was showcased in the comparisons of compaction methods made in [4].

Despite powerful, this technique provides edge-length optimum results only for 4-planar graphs. This is unfortunate, given that graphs in real-life applications often have higher vertex degree.

Some methods have proposed ways of representing and describing graphs with vertex degree greater than four. Among them, the Kandinsky model [2] takes center stage. It allows edges to run on a finer grid, which means that, without altering the size of vertices, many edges may be incident in parallel on the same side. In comparison to other approaches [6, 7, 9], it is space-efficient and does not abandon the expected orthogonality of the drawings.

In this work, we unite these two elements into a single procedure. Specifically, we propose an extension of Klau and Mutzel's compaction method [5], such that any planar graph may be optimally compacted and ultimately drawn according to the Kandinsky model, regardless of its vertex degree.

In order to do this, the properties presented by Klau and Mutzel [5] to define the set of feasible solutions - in particular, *separation* and *adjacency* - for 4-planar graphs are extended to cover previously inexistent situations. In addition, novel concepts - *aggregating/twin segments* - are introduced to ensure correctness and optimality. Concretely, *aggregating segments* are the result of transforming parallel edges - as the ones typical of the Kandinsky model - into equivalent, as far as the optimization problem is concerned, sequences of dummy vertices and edges. When parallel edges bend both to the left and to the right, then it is necessary to introduce *twin segments*, which are nothing more than two aggregating segments that must occupy the same grid position and represent different layout constraints.

The obtained results shed light on two relevant points: (i) from a practical perspective, being able to optimally compact and draw absolutely any planar graph is a relevant feature in many systems; (ii) computationally, it has been observed that the increased cardinality of the graph (and, thus, average vertex degree) does not compromise the overall execution time, due to the preprocessing stage presented in [5] being more efficient for graphs with relatively many faces.

Therefore, it is our belief that this methodology is both of theoretical interest and highly applicable in practice.

Acknowledgment. The authors would like to thank CNPq, Brazil for supporting this research.

References

1. Eiglsperger, M., Kaufmann, M.: Fast compaction for orthogonal drawings with vertices of prescribed size. In: Mutzel, P., Jünger, M., Leipert, S. (eds.) GD 2001. LNCS, vol. 2265, pp. 124–138. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45848-4_11
2. Fößmeier, U., Kaufmann, M.: Drawing high degree graphs with low bend numbers. In: Brandenburg, F.J. (ed.) GD 1995. LNCS, vol. 1027, pp. 254–266. Springer, Heidelberg (1996). <https://doi.org/10.1007/BFb0021809>
3. Klau, G.W., Mutzel, P.: Combining graph labeling and compaction. In: Kratochvířl, J. (ed.) GD 1999. LNCS, vol. 1731. Springer, Heidelberg (1999)
4. Klau, G.W., Klein, K., Mutzel, P.: An experimental comparison of orthogonal compaction algorithms. In: Marks, J. (ed.) GD 2000. LNCS, vol. 1984, pp. 37–51. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44541-2_5
5. Klau, G.W., Mutzel, P.: Optimal compaction of orthogonal grid drawings (extended abstract). In: Cornuéjols, G., Burkard, R.E., Woeginger, G.J. (eds.) IPCO 1999. LNCS, vol. 1610, pp. 304–319. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48777-8_23
6. Klau, G., Mutzel, P.: Quasi-orthogonal Drawing of Planar Graphs. Bibliothek & Dokumentation, MPI Informatik (1998)

7. Otten, R.H.J.M., van Wijk, J.G.: Graph representations in interactive layout design. In: Proceedings of the IEEE International Symposium. on Circuits and Systems, pp. 914–918 (1978)
8. Patrignani, M.: On the complexity of orthogonal compaction. *Comput. Geom.* **19**(1), 47–67 (2001)
9. Tamassia, R., Battista, G.D., Batini, C.: Automatic graph drawing and readability of diagrams. *IEEE Trans. Syst. Man Cybern.* 61–79 (1988)

BCSA: BC Tree-Based Sampling and Visualization of Big Graphs

Seok-Hee Hong^(✉), Quan Nguyen, Amyra Meidiana, and Jiayi Li

The School of Information Technologies, University of Sydney, Sydney, Australia
{seokhee.hong, quan.nguyen}@sydney.edu.au,
{amei2916, jili2506}@uni.sydney.edu.au

Abstract. Graph sampling techniques have been popular for the analysis and visualization of big complex networks. However, existing sampling methods often fail to preserve connectivity and important global skeletal structure in the original graph. This poster introduces two new families of sampling methods **BCSA-W** and **BCSA-E** for big complex graphs, based on the decomposition of a graph into biconnected components, known as the BC (Block Cut-vertex) tree. Experimental results using graph sampling quality metrics show that our new sampling methods produce better results than existing methods: 25% improvement by **BCSA-W** and 15% by **BCSA-E** over existing methods on average.

We then present **DBCSEA**, BC tree based graph sampling algorithm in distributed environment. Experiments on the Amazon Cloud EC2 demonstrate that **DBCSEA** is scalable for big graph data sets; running time speed up of 77% for distributed 5-server sampling over sequential sampling on average. We also present a new layout method called **BCTV**, which clearly shows the BC tree decomposition of a graph. Visual comparison using the **BCTV** layout shows that our new sampling methods can better maintain the global structure of the original graph.

1 Introduction

Graph sampling has been well studied in graph mining for analysis of big graphs [1]. A number of graph sampling methods have been proposed and evaluated using various quality metrics [1]. Graph sampling has also been used for the visualization of large and complex networks [2]. Recently, Zhang *et al.* presented experimental comparison of different sampling algorithms under various sampling metrics [4]. More recently, Wu *et al.* presented user studies to investigate how sampling methods influence graph visualization, in terms of human perception of high degree vertices, clusters and coverage area [3].

Graph sampling methods commonly aim to retain some structural properties of the original graph; however, they often fail to preserve connectivity as well as the important global skeletal structure of the graph. In particular, Random Vertex and Random Edge sampling often produce a set of disconnected subgraphs as samples [3].

In this poster, we propose a new approach for sampling big complex graphs, called **BCSA** (**BC** tree-based **SA**mpling), based on the decomposition of a graph into *biconnected components*, called *BC (Block Cut-vertex) tree*, to maintain the connectivity structure and the important global skeletal structure of the original graph.

More specifically, we first include the *cut vertices* of the original graph to graph samples, since cut vertices are structurally important vertices in terms of connectivity as well as network analysis. For example, in social network analysis, cut vertices are important actors of the network, since their social roles and positions are brokers or actors connecting different communities, often with high betweenness centrality.

The main contribution of this poster can be described as follows:

1. We introduce two new families of sampling methods **BCSA-W** (BCSA-Whole) and **BCSA-E** (BCSA-Each). Each family consists of five different sampling methods, based on five most popular sampling methods, combined with the BC tree decomposition. **BCSA-W** algorithms first add cut vertices to the samples, and then perform sampling in a similar way as the original sampling algorithms. **BCSA-E** is a Divide and Conquer algorithm which performs **BCSA-W** algorithms for each biconnected component independently and then merges the results to obtain the final sample.

Experimental results with real world graph data sets demonstrate that **BCSA** algorithms produce better quality samples than the corresponding original sampling methods, using well-known sampling quality metrics; 25% improvement was obtained by **BCSA-W** and 15% by **BCSA-E** over existing methods on average.

2. We present **DBC**SA (**D**istributed **BC** tree-based **SA**mpling) to exploit the BC tree decomposition in distributed environment for sampling big complex network efficiently. The BC tree decomposition gives a set of biconnected components, overlapping (i.e., sharing) only at cut vertices. Such a decomposition is therefore useful for effectively partitioning the set of biconnected components to a set of servers, and to reduce communication overhead on a distributed computing platform. More specifically, we use the Divide and Conquer **BCSA-E** algorithms for sampling biconnected components in each server in parallel.

Experiments on the Amazon Cloud EC2 with both real world graph data sets and synthetic data sets demonstrate that **DBC**SA is scalable for big graph data sets; on average, the running time speed up is 77% on distributed 5-server, and 46% on 2-server over sequential sampling.

3. For better visual comparison of the sampling results, we present a new graph layout algorithm **BCTV** (**BC** Tree Visualization), which clearly shows the BC tree decomposition of a graph. More specifically, **BCTV** is a divide and conquer algorithm, which combines a weighted tree drawing algorithm (for drawing the BC tree) and a force-directed algorithm (for drawing each biconnected component).

Visual comparison using BCTV layout with real world data sets visually confirm that our new sampling methods can better maintain the global structure of the original graph.

References

1. Hu, P., Lau, W.C.: A survey and taxonomy of graph sampling. CoRR abs/1308.5865 (2013)
2. Rafei, D., Curial, S.: Effectively visualizing large networks through sampling. In: 16th IEEE Visualization Conference, VIS 2005, Minneapolis, MN, USA, 23–28 October 2005, pp. 375–382 (2005)
3. Wu, Y., Cao, N., Archambault, D., Shen, Q., Qu, H., Cui, W.: Evaluation of graph sampling: a visualization perspective. *IEEE Trans. Vis. Comput. Graph.* **23**(1), 401–410 (2017)
4. Zhang, F., Zhang, S., Wong, P.C., Swan II, J.E., Jankun-Kelly, T.: A visual and statistical benchmark for graph sampling methods. In: Exploring Graphs at Scale (EGAS) Workshop, IEEE VIS 2015, October 2015

Low Ply Drawings of Trees of Bounded Degree

Michael T. Goodrich and Timothy Johnson^(✉)

Department of Computer Science, University of California, Irvine, CA, USA

1 Introduction

An interesting paradigm for drawing graphs involves visualizing them as maps or road networks, allowing a visualizer to use known techniques that apply to maps.

Eppstein and Goodrich [3] introduce the concept of *ply number* of an embedded graph and they demonstrate experimentally that real-world road networks tend to have low ply. Intuitively, the ply concept tries to capture the way that road networks have features that are well-separated at multiple scales.

The ply number of a drawing is computed by first assigning to each vertex a disk with a radius of α times the length of its longest incident edge. The ply number is then the maximum number of disks that intersect in a single point.

2 Our Results

We announce two new results in drawing graphs with low ply number. Di Giacomo *et al.* [2], asked whether all bounded-degree trees have 1-ply drawings for a sufficiently small α . We show that this is indeed the case.

We then extend a result from Angelini *et al.* [1], who showed that trees with maximum degree six can be drawn in polynomial area with logarithmic ply number. We extend this as well to trees of any bounded degree, using the heavy-path decomposition technique of Sleator and Tarjan [4].

3 1-Ply Drawings

At a high level, our 1-ply drawings of bounded-degree trees are constructed as follows. For a tree with maximum degree Δ , we divide the area around each vertex radially into Δ equal wedges, each containing one of its neighbors. The distance from each node to its children is chosen to be a constant fraction f of its distance from its own parent.

The following three constraints ensure that the ply number is 1. Comparing these constraints then allows us to prove the following theorem.

1. Ply disks for adjacent vertices must not overlap.
2. Ply disks for vertices on separate subtrees must not overlap.
3. A ply disk for a vertex must never overlap a ply disk for one of its ancestors.

Theorem 1. *Let T be a tree with maximum degree Δ , and let $f = \frac{\sin(\frac{\pi}{\Delta})}{1 + \sin(\frac{\pi}{\Delta})}$. T has a 1-ply drawing if $\alpha \leq \min\{\frac{f}{1+f}, f\sqrt{1 - 2f \cos \theta + f^2} - \frac{f^3}{1-f}\}$.*

4 Polynomial Area, Logarithmic Ply Number

Theorem 2. *For $\alpha = 0.5$, a tree with maximum degree Δ can be drawn with ply number $O(\log n)$ in area $n^{O(\Delta)}$.*

We first describe a simple algorithm for drawing trees by layering their children, which proves this theorem for balanced trees. This relies on the following lemma.

Lemma 1. *Suppose that r is the root of a star graph. Let v_1 and v_2 be children at distances d_1 and d_2 , respectively. If $d_2 \geq 3d_1$, then the ply disks for v_1 and v_2 are disjoint.*

Next we show that, as in Angelini et. al. [1], the heavy path decomposition will allow us to draw these trees with logarithmic ply number even when they are unbalanced. We let $\mu = (v_1, v_2, \dots, v_m)$ be a path in our heavy path decomposition, a_μ be its anchor vertex, n_μ is the number of vertices whose paths are anchored at a_μ , n_i is the number of vertices whose paths are anchored at v_i , and $l(a, b)$ is the length of an edge from a to b . We use the algorithm DRAWPATH from Angelini et. al. [1], which satisfies the following property.

Lemma 2. *Algorithm DRAWPATH constructs a drawing Γ with ply number 2 of a path $\mu = (v_1, v_2, \dots, v_m)$ such that $l(a_\mu, v_1) \geq n_1$, $l(v_i, v_{i+1}) \geq n_i + n_{i+1}$, for each $i = 1, \dots, m - 1$, and $l(\Gamma) \leq 6n_\mu$.*

We then use this DRAWPATH algorithm to perform a bottom-up construction of our tree, in which the child paths of each vertex in our tree are assigned to different layers around their root. We space each layer such that each node in layer $i + 1$ is at least three times the distance of any node in layer i , so that none of the ply disks for nodes in different layers overlap.

We then scale each path so that the child paths of any two adjacent vertices in a path do not intersect. This produces a drawing with a ply number of $3(h + 1)$, where h is the height of the heavy path decomposition. The spacing provided by the DRAWPATH algorithm ensures that our scaling factor is constant for each level of the decomposition, which implies that the total area is polynomial.

References

1. Angelini, P., Bekos, M.A., Bruckdorfer, T., Hančl, J., Kaufmann, M., Kobourov, S., Symvonis, A., Valtr, P.: Low ply drawings of trees. In: Hu, Y., Nöllenburg, M. (eds.) GD 2016. LNCS, vol. 9801, pp. 236–248. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_19
2. Di Giacomo, E., Didimo, W., Hong, S.H., Kaufmann, M., Kobourov, S.G., Liotta, G., Misue, K., Symvonis, A., Yen, H.C.: Low ply graph drawing. In: 2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA), pp. 1–6. IEEE (2015)
3. Eppstein, D., Goodrich, M.T.: Studying (non-planar) road networks through an algorithmic lens. In: Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, p. 16. ACM (2008)
4. Sleator, D.D., Tarjan, R.E.: A data structure for dynamic trees. *J. Comput. Syst. Sci.* **26**(3), 362–391 (1983)

Which Graph Layout Gives a Good Shape for Large Graphs?

Quan Nguyen^(✉), Peter Eades, and Seok-Hee Hong

The School of Information Technologies, University of Sydney, Sydney, Australia
{quan.nguyen,peter.eades,seokhee.hong}@sydney.edu.au

Abstract. This poster empirically investigates the quality of large graph layouts using shape-based quality metrics. We present our preliminary results on several real-world graphs.

1 Introduction

Drawing very large graphs is challenging and has attracted extensive research. Force directed methods are, so far, the most popular methods used for drawing large graphs.

Traditionally, graph drawing algorithms often aim for some aesthetic criteria; for example, edge bends, edge crossings and angular resolution; these metrics are good for small graphs of up to a few hundred nodes [13].

Shape-based faithfulness metrics [7] measure how well the “shape” of the drawing represents the graph. The drawing D is *shape faithful* if one can derive G from the shape of D . For large graphs, the shape of the drawing is more significant than the number of edge bends and edge crossings. Given a graph G and P is the set of vertex locations in the drawing D of G . A *shape graph* $S(P)$ is a graph with vertex set P such that the edges of $S(P)$ form the “shape” of P . Examples of shape graphs are the *Euclidean minimum spanning tree (EMST)*, the *relative neighbourhood graph (RNG)*, and the *Gabriel graph (GG)* [16]. The *shape-based quality* of a drawing D is defined as the similarity between the shape graph $S(P)$ and G .

In this poster, we investigate the quality and speed of a number of standard large graph layout algorithms. For an objective evaluation, we use the shape-based quality metrics [7]. We empirically investigated a number of real-world graphs. The graphs contain thousands of vertices and tens of thousands of edges. Our results lead to suggestions of which algorithms are better than others.

2 Experiment

Data Sets. For our evaluation, we used the graphs taken from the Hachul library, Walshaw’s Graph Partitioning Archive, the sparse matrices collection [6] and the network repository [15]. These include two commonplace types of

graphs that have been extensively studied in graph drawing research: grid-like graphs and scale-free graphs.

Layouts. For layout, we used available implementations of standard force-directed algorithms implemented in OGDF [4]. We also used the *LinLog* layout [14], Tom Sawyer’s *symmetric* force-directed layout [1] and yEd’s *Organic* layout [2].

Particularly, we used the following standard force-directed algorithms that are available in OGDF: (1) *FM³* by Hachul and Junger [11]; (2) *stress* [10]; (3) *FR* by Fruchterman and Reingold [9]; (4) *PivotMDS* by Brandes and Pich [3]; (5) *KK* by Kamada and Kawai [12]. We also used variations of force-directed layouts available in OGDF. They include (i) *FMME*; (ii) *Nice*; (iii) *Fast*; (iv) *NoTwist*; (v) *MixedForce* and (vi) *FRGrid*: a grid-variant of the FR algorithm [9]. In addition, we also tried *GEM* layout by Frick et al. [8], and *DH* layout by Davidson and Harel [5]; however, they did not scale.

In our experiments, we run each layout algorithm for each data set using the default values of the parameters.

Settings. The experiments were performed on a i7 XPS Dell laptop with 16 GB memory and 512 GB SSD running Ubuntu 16.04. The runtime was capped to 15 min for each algorithm on each graph. OGDF’s layout implementations were compiled with gcc 5.4.0, x86-64 (-O3). The evaluation of LinLog was conducted on the same machine with Java 8 and 6 GB of heap.

Quality Results. Overall, Fast, Nice and NoTwist are the winners for all graphs. Tom Sawyer performed quite well, giving good shapes for most of the graphs. FMME gave good results for many graphs, except for yeastppi, oflights and p2p-Gnutella05 data sets. LinLog performed quite well for scale-free graphs, but gave low quality for grid-like graphs. FM³, yEd, stress and PMDS belong to the average performers. Layouts from FR, DH and GEM layouts using default parameters are not so good.

Running Time. Stress, NoTwist, KK and LinLog consumed more time than the other algorithms. DH finished only for one graph (can144) within 15 min.

References

1. Tom Sawyer Software: Graph layout toolkit. <http://www.tomsawyer.com>
2. yEd - Java Graph Editor. <http://www.yworks.com/products/yed>
3. Brandes, U., Pich, C.: Eigensolver methods for progressive multidimensional scaling of large data. In: Kaufmann, M., Wagner, D. (eds.) GD 2006. LNCS, vol. 4372, pp. 42–53. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70904-6_6
4. Chimani, M., Gutwenger, C., Jünger, M., Klau, G.W., Klein, K., Mutzel, P.: The Open Graph Drawing Framework (OGDF). CRC Press (2012)
5. Davidson, R., Harel, D.: Drawing graphs nicely using simulated annealing. *ACM Trans. Graph.* **15**(4), 301–331 (1996)
6. Davis, T.A., Hu, Y.: The university of florida sparse matrix collection. *ACM Trans. Math. Softw.* **38**(1), 1:1–1:25 (2011)

7. Eades, P., Hong, S., Nguyen, A., Klein, K.: Shape-based quality metrics for large graph visualization. *J. Graph Algorithms Appl.* **21**(1), 29–53 (2017)
8. Frick, A., Ludwig, A., Mehldau, H.: A fast adaptive layout algorithm for undirected graphs. In: Tamassia, R., Tollis, I.G. (eds.) GD 1994. LNCS, vol. 894, pp. 388–403. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-58950-3_393
9. Fruchterman, T., Reingold, E.: Graph drawing by force-directed placement. *Softw. Pract. Experience* **21**(11), 1129–1164 (1991)
10. Gansner, E.R., Koren, Y., North, S.: Graph drawing by stress majorization. In: Pach, J. (ed.) GD 2004. LNCS, vol. 3383, pp. 239–250. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31843-9_25
11. Hachul, S., Jünger, M.: Drawing large graphs with a potential-field-based multilevel algorithm. In: Pach, J. (ed.) GD 2004. LNCS, vol. 3383, pp. 285–295. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31843-9_29
12. Kamada, T., Kawai, S.: An algorithm for drawing general undirected graphs. *Inf. Process. Lett.* (1989)
13. Nguyen, Q.H., Eades, P., Hong, S.: On the faithfulness of graph visualizations. In: IEEE Pacific Visualization Symposium, PacificVis 2013, 27 February–1 March 2013, Sydney, NSW, Australia, pp. 209–216 (2013)
14. Noack, A.: Energy models for graph clustering. *J. Graph Algorithms Appl.* **11**(2), 453–480 (2007)
15. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: AAAI (2015). <http://networkrepository.com>
16. Toussaint, G.T.: The relative neighbourhood graph of a finite planar set. *Pattern Recognit.* **12**(4), 261–268 (1980)

MetagenomeScope

Web-Based Hierarchical Visualization of Metagenome Assembly Graphs

Marcus Fedarko^{1,2}, Jay Ghurye^{1,2}, Todd Treangen², and Mihai Pop^{1,2}(✉)

¹ Department of Computer Science, University of Maryland, College Park, USA
jayg@cs.umd.edu, mpop@umiacs.umd.edu

² Center for Bioinformatics and Computational Biology, University of Maryland,
College Park, USA
{mfedarko,treangen}@umd.edu

1 Introduction

Sequencing technologies break up DNA into many small fragments, necessitating the reconstruction of these fragments in order to identify the complete structure of the input sequence. This process is referred to as assembly. In particular, *genome assembly* involves assembling sequences arising from a single organism's DNA, while *metagenome assembly* involves assembling sequences taken from uncultured environments in which many organisms' genetic material may be contained.

Most assemblers represent the fragments of DNA resulting from sequencing as an overlap graph or a de Bruijn graph [5]. In either type of graph, nodes correspond to pieces of DNA obtained from these fragments (also referred to as *contigs*) and edges correspond to overlaps between contigs' sequences. The ideal graph obtained from assembly would consist of c connected components comprised of single linear or circular paths, where c is the number of DNA sequences in the input sample. In turn, these paths would spell out the original sequence(s). However, complexities such as repetitive sequences and sequencing errors can create branches and cycles in the graph—so the graphs generated by assemblers often require manual examination after the fact to resolve ambiguous connections and correct assembly errors [8]. This has brought about the need for tools that can visualize these frequently-intricate graphs effectively, displaying relevant biological metadata and graph structural information in a readily accessible manner. Furthermore, there is a documented dearth of hierarchical visualization tools that allow the user to go “from the large structure down to the base level [of the assembly graph]” [6].

To help resolve this deficit we present MetagenomeScope, an interactive web-based tool for the visualization of assembly graphs. MetagenomeScope contains a number of features to aid bioinformaticians in exploratory analysis of these structures at both coarse and fine levels of complexity.

2 Primary Contributions

MetagenomeScope uses the *dot* tool for hierarchical layout [3] to draw graphs in its “standard mode.” This differs notably from existing assembly visualization tools such as Bandage [9], ABySS-Explorer [7], and Ray Cloud Browser [4], all of which employ force-directed layouts when drawing graphs. We found that force-directed drawings of assembly graphs tended to produce visualizations for which manual inspection of the graph’s structural details, such as “bubble” patterns [5], became onerous.

To further emphasize certain biologically relevant structural patterns in the graph, we modified MetagenomeScope’s layout process to group together collections of contigs identified as belonging to these patterns. Along with being clustered together during layout, these contig groups can also be dynamically collapsed and uncollapsed by the user, allowing the user to reduce and expand the scope of the graph.

MetagenomeScope also supports the use of SPQR trees [2] as a means for decomposing biconnected components in the graph into simpler, iteratively expandable regions. This is employed in MetagenomeScope’s SPQR “decomposition mode.” We currently generate an undirected version of the input graph in which each biconnected component is replaced with a “node” containing the root metanode of its corresponding SPQR tree. SPQR trees are computed using the Open Graph Drawing Framework [1]. In MetagenomeScope’s viewer interface, the metanodes in the tree can be iteratively expanded to reveal further details about the paths through their respective biconnected component. Along with helping the user trace through paths in biconnected components, this feature yields a greatly simplified initial view of the overall assembly graph. SPQR trees have been recognized as a potential means for “finding the tangles in the [assembly] graph” [6], and we plan to continue developing this feature as we receive feedback on MetagenomeScope.

3 Demonstrated Applications

The visualizations provided by MetagenomeScope have already seen practical use in our lab. Scaffolds (oriented and ordered groups of adjacent contigs) defined in AGP files can be visualized in MetagenomeScope as overlaid on top of their respective contigs in the assembly graph; we have used this feature to visually identify nonadjacent scaffolds, thus indicating an error in our scaffold-generating code. Upon fixing the code in question, we were able to use MetagenomeScope to verify that the scaffolds generated for the graph in question were properly redesigned.

The visualization component of the tool’s status as a web application has also been of use, since it effectively minimizes the software and hardware requirements for viewing assembly graphs. We have already used MetagenomeScope’s viewer interface to display assembly graphs as part of the 2017 “Strategies and Techniques for Analyzing Microbial Population Structure” research training course held at the Marine Biological Laboratory.

Acknowledgements. The authors were supported in part by the NIH, grant R01-AI-100947, the NSF, grant IIS-1117247NRL, and the Navy Research Laboratories, cooperative agreement N00173162C001, all to MP.

References

1. Chimani, M., Gutwenger, C., Jünger, M., Klau, G.W., Klein, K., Mutzel, P.: The open graph drawing framework (OGDF). In: Tamassia, R. (ed.) *Handbook of Graph Drawing and Visualization*. CRC Press (2014)
2. Di Battista, G., Tamassia, R.: Incremental planarity testing. In: *30th Annual Symposium on foundations of Computer Science*, pp. 436–441. IEEE (1989)
3. Ellson, J., Gansner, E., Koutsofios, L., North, S.C., Woodhull, G.: Graphviz — open source graph drawing tools. In: Mutzel, P., Jünger, M., Leipert, S. (eds.) *GD 2001*. LNCS, vol. 2265, pp. 483–484. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45848-4_57
4. Godzaridis, E., Boisvert, S., Xia, F., Kandel, M., Behling, S., Long, B., Sosa, C.P., Laviolette, F., Corbeil, J.: Human analysts at superhuman scales: what has friendly software to do? *Big Data* **1**(4), 227–236 (2013)
5. Miller, J.R., Koren, S., Sutton, G.: Assembly algorithms for next-generation sequencing data. *Genomics* **95**(6), 315–327 (2010)
6. Myers, G., Pop, M., Reinert, K., Warnow, T.: Next generation sequencing (Dagstuhl seminar 16351). *Dagstuhl Rep.* **6**(8), 91–130 (2017)
7. Nielsen, C.B., Jackman, S.D., Birol, I., Jones, S.J.M.: Abyss-explorer: visualizing genome sequence assemblies. *IEEE Trans. Visual. Comput. Graph.* **15**(6), 881–888 (2009)
8. Phillippy, A.M., Schatz, M.C., Pop, M.: Genome assembly forensics: finding the elusive mis-assembly. *Genome Biol.* **9**(3), R55 (2008)
9. Wick, R.R., Schultz, M.B., Zobel, J., Holt, K.E.: Bandage: interactive visualization of de novo genome assemblies. *Bioinformatics* **31**(20), 3350–3352 (2015)

Author Index

- Akitaya, Hugo A. 210, 615
Angelini, Patrizio 24, 102, 517
Archambault, Daniel 394
Arleo, Alessio 256
- Bae, Sang Won 531
Baffier, Jean-Francois 531
Ballweg, K. 241
Bekos, Michael A. 102, 169, 426, 517
Biedl, Therese 81, 140, 184, 305, 318, 600
Bishnu, Arijit 603
- Calamoneri, Tiziana 334
Chan, Timothy M. 305
Chaplick, Steven 24, 465, 546
Chimani, Markus 465
Chun, Jinhee 531
Cornelsen, Sabine 465
- Da Lozzo, Giordano 465
De Luca, Felice 24, 426
Demaine, Erik D. 210
Derka, Martin 184, 305
Devanny, William 575
Di Donato, Valentino 334
Di Giacomo, Emilio 413, 479, 609
Didimo, Walter 256, 426
Dujmović, Vida 184
- Eades, Peter 272, 531, 627
Eickmeyer, Kord 531
Eppstein, David 506, 560
- Fedarko, Marcus 630
Felsner, Stefan 127
Fiala, Jiří 24
Förster, Henry 169
Fröschl, Theresa 585
Fulek, Radoslav 160, 594
- Gasieniec, Leszek 413
Ghosh, Arijit 603
Ghurye, Jay 630
- Gimbel, John 67
Goodrich, Michael T. 624
Grilli, Luca 531
- Hančl Jr., Jaroslav 24
Heinsohn, Niklas 24, 38
Hesterberg, Adam 210
Hong, Seok-Hee 272, 531, 621, 627
- Irvine, Veronika 140
- Jain, Kshitij 305
Jianu, Radu 287
Johnson, Timothy 624
Jones, Matthew D. 615
- Kaufmann, Michael 24, 38, 102, 169, 517
Kindermann, Philipp 52, 113, 517, 575
Kirby, Robert M. 606
Klawitter, Jonathan 224
Klemz, Boris 440
Klute, Fabian 612
Kobourov, Stephen 24, 113, 287, 394
Korman, Matias 531
Kostitsyna, Irina 492
Kratochvíl, Jan 24
Kryven, Myroslav 546, 591
Kynčl, Jan 594
- Lazard, Sylvain 17
Lenhart, William 17
Li, Jiaxi 621
Liotta, Giuseppe 17, 256, 413, 479, 546, 609
Lipp, Fabian 365
Lisboa, Adriano Chaves 618
Liu, Quanquan C. 210
Löffler, Andre 546
Löffler, Maarten 113, 575
Lubiw, Anna 305
- Mariottini, Diego 334
Markfelder, Peter 365
Mathew, Rogers 603

- Mchedlidze, Tamara 3, 224, 426
Mehrabi, Saeed 81
Meidiana, Amyra 621
Meulemans, Wouter 52, 492
Miller, Nathaniel 588
Mishra, Gopinath 603
Mondal, Debajyoti 318, 600
Montecchiani, Fabrizio 102, 256, 531
Morin, Pat 184
- Navarra, Alfredo 413
Nguyen, Quan 272, 621, 627
Nöllenburg, Martin 113, 224, 426, 465, 585, 612
- Oikonomou, Anargyros 326
Okoe, Mershack 287
Ossona de Mendez, Patrice 67
- Pach, János 153, 160
Patrignani, Maurizio 334, 465, 479, 609
Paul, Subhabrata 603
Pohl, M. 241
Pop, Mihai 630
Pupyrev, Sergey 197
- Radermacher, Marcel 3, 597
Raj, Mukund 351, 606
Ramos, Eduardo Santiago 618
Ravsky, Alexander 591
Reichard, Klara 597
Rote, Günter 440
Ruginski, Ian 606
Rutter, Ignaz 3, 531, 575, 597
- Sandoval, Gregory A. 615
Scheucher, Manfred 127
- Schneck, Thomas 517
Schulz, André 52, 113
Silveira, Rodrigo I. 454
Simonetto, Paolo 394
Sitchinava, Nodari 88
Sondag, Max 492
Souvaine, Diane L. 615
Speckmann, Bettina 454
Stalfa, David 615
Strash, Darren 88
Symvonis, Antonios 326, 426
- Tappini, Alessandra 479, 609
Tollis, Ioannis G. 426, 465
Tóth, Csaba D. 531, 615
Tóth, Géza 153
Treangen, Todd 630
- Valtr, Pavel 24, 67
van Dijk, Thomas C. 365
van Goethem, Arthur 492
van Kreveld, Marc 492
Verbeek, Kevin 454
Vogtenhuber, Birgit 113
von Landesberger, T. 241
- Wagner, Dorothea 597
Wallner, G. 241
Whitaker, Ross T. 351, 606
Wolff, Alexander 365, 465, 546, 591
Wu, Jieting 379
Wulms, Jules 492
- Yu, Hongfeng 379
- Zeng, Jianping 379
Zhu, Feiyu 379