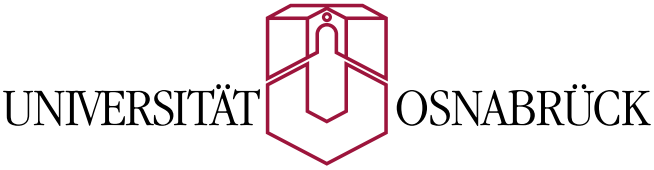
Computing Measures of Non-Planarity

Dissertation

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

im Fachbereich Mathematik/Informatik an der



vorgelegt von

Tilo Wiedera

Osnabrück

2021

Dekan
Prof. Dr. Tim Römer

Gutachter
Prof. Dr. Markus Chimani, Universität Osnabrück
Prof. Dr. Ignaz Rutter, Universität Passau

Zusammenfassung

Planare Graphen haben eine mannigfaltige Geschichte deren Anfänge sich bis in das 18. Jahrhundert zurückverfolgen lassen. Sie bilden eines der grundlegenden Konzepte der Graphentheorie. In der rechentechnischen Graphentheorie bietet die Planarität von Graphen signifikante Vorteile in Hinblick auf den Algorithmenentwurf und viele bahnbrechende algorithmischen Ergebnisse für planare Graphen wurden im Laufe der Zeit entdeckt. Formal gesehen ist ein Graph entweder planar oder nicht(-planar). Jedoch existiert eine vielfältige Menge von etablierten Maßen um zu messen wie weit ein Graph davon entfernt ist, planar zu sein.

In der vorliegenden Arbeit beschäftigen wir uns mit der Evaluation und Verbesserung von Algorithmen, die diese Maße der Nicht-Planarität berechnen. Insbesondere betrachten wir (1) das Problem einen planaren Teilgraphen mit maximaler Kantenanzahl zu finden, (2) das Problem einen Graphen auf einer Oberfläche mit möglichst kleinem Genus einzubetten und (3) das Problem einen Graphen mit möglichst wenig Kantenkreuzungen zu zeichnen. All diese Probleme sind klassische Fragen des Graphenzeichnens und jeweils NP-schwer. Dennoch gibt es teilweise jahrzehntelange Forschung über exakte Algorithmen zur Berechnung dieser Maße. Wir präsentieren neue Modelle, basierend auf mathematischer Programmierung und verschiedenen Charakterisierungen von Planarität, um große planare Teilgraphen und Einbettungen mit kleinem Genus zu finden. Die Quintessenz unserer erfolgreichsten Modelle ist es auch die Beziehung zwischen Einbettungen und ihrem dualen Graph angemessen genau zu beschreiben. Basierend auf diesen Modellen entwerfen wir neue Algorihmen die um Größenordnungen schneller als der aktuelle Stand der Technik sind. Wir untermauern diese Beobachtungen durch umfassende experimentelle Studien und zeigen außerdem die theoretischen Vorteile unserer neuen Modelle auf.

Neben exakten Algorithmen betrachten wir auch Heuristiken und approximative Verfahren um große planare Teilgraphen zu berechnen. Im Bereich der Kreuzungszahlen präsentieren wir eine automatische Beweisextraktion, ein neues Härteresultat in Bezug auf das Einfügen einzelner Kanten sowie die Lösung der verbleibenden Fälle einer Vermutung über den Zusammenhang zwischen hohem Knotengrad und minimalen Hindernissen für niedrige Kreuzungszahl.

Abstract

Planar graphs have a rich history that dates back to the 18th Century. They form one of the core concepts of graph theory. In computational graph theory, they offer broad advantages to algorithm design and many groundbreaking results are based on them. Formally, a given graph is either planar or non-planar. However, there exists a diverse set of established measures to estimate how far away from being planar any given graph is.

In this thesis, we aim at evaluating and improving algorithms to compute these measures of non-planarity. Particularly, we study (1) the problem of finding a maximum planar subgraph, i.e., a planar subgraph with the least number of edges removed; (2) the problem of embedding a graph on a lowest possible genus surface; and finally (3) the problem of drawing a graph such that there are as few edge crossings as possible. These problems constitute classical questions studied in graph drawing and each of them is NP-hard. Still, exact (exponential time) algorithms for them are of high interest and have been subject to study for decades. We propose novel mathematical programming models, based on different planarity criteria, to compute maximum planar subgraphs and low-genus embeddings. The key aspect of our most successful new models is that they carefully describe also the relation between embedded (sub-)graphs and their duals. Based on these models, we design algorithms that beat the respective state-of-the-art by orders of magnitude. We back these claims by extensive computational studies and rigorously show the theoretical advantages of our new models.

Besides exact algorithms, we consider heuristic and approximate approaches to the maximum planar subgraph problem. Furthermore, in the realm of crossing numbers, we present an automated proof extraction to easily verify the crossing number of any given graph; a new hardness result for a subproblem that arises, e.g., when enumerating simple drawings; and resolve a conjecture regarding high node degree in minimal obstructions for low crossing number.

Acknowledgments

I want to express my deep gratitude to everyone who helped me in one way or another with my research and thesis. I would particularly like to thank my external reviewer Ignaz Rutter for straightforwardly agreeing to evaluate this thesis. Naturally, I would like to sincerely thank my advisor Markus Chimani for sparking my late interest in theoretical computer science and graph theory. Markus, without your genuine enthusiasm and thorough knowledge, I would probably have missed the great endeavor that scientific research is. Indeed, I am grateful for the help of many colleagues: During my first years, Stephan Beyer and Ivo Hedtke eagerly answered my many questions and were always open to the discussion of research ideas. Niklas Troost, I am glad that we found a common interest in multistage problems and hope to continue along this path. Fritz Bökler, I appreciate our discussions and the clarity that you are able to express even when considering complex matters. Alexander Nover, you are a great office neighbor and I will surely miss your commentary! Clearly, I am also indebted to our many student assistants who worked so tirelessly on testing and implementing parts of our software.

During my travels I met many wonderful people and great scientists. I particularly enjoyed the outstanding annual *Crossing Numbers Workshop* that provides a prime atmosphere for collaborative research. It brought much joy to me and I sincerely hope to participate in some of its future editions.

Of course, I would like to extend my gratitude also to my many co-authors. You are fantastic and working with you was—and still is—a delight!

I am much obliged also to the Deutsche Forschungsgemeinschaft and Osnabrück University for funding my research.

Finally, I would like to thank my family and friends for their uncompromising support—even (and particularly) for your firm reminders about the importance of leisure and occasional vacations.

Contents

1 Introduction **1**
- 1.1 Basic Notation . . . 2
- 1.2 Complexity Theory . . . 2
- 1.3 Graph Theory . . . 5
- 1.4 Planarity . . . 9
- 1.5 Measures of Non-Planarity . . . 13
  - 1.5.1 Skewness and Maximum Planar Subgraph . . . 15
  - 1.5.2 Vertex Deletion Number and MPIS . . . 16
  - 1.5.3 Splitting Number . . . 17
  - 1.5.4 Genus . . . 17
  - 1.5.5 Crossing Number . . . 18
  - 1.5.6 Thickness . . . 19
  - 1.5.7 Coarseness . . . 19
- 1.6 Integer Linear Programming . . . 21
- 1.7 Experiments and Algorithm Engineering . . . 23
  - 1.7.1 Instance Sets . . . 24

2 Skewness and MPS **27**
- 2.1 ILP Model by Mutzel . . . 27
- 2.2 Heuristics . . . 28
  - 2.2.1 Algorithms . . . 30
  - 2.2.2 Experiments . . . 32
- 2.3 Approximation of MPS and its Limits . . . 34
  - 2.3.1 Algorithms Inspired by Planarity Tests . . . 37
  - 2.3.2 MPS is NP-hard: A Simple Proof . . . 42
  - 2.3.3 Algorithms Inspired by Cactus Structures . . . 43
- 2.4 ILP Models by Planarity Criteria . . . 49
  - 2.4.1 Facial Walks . . . 50
  - 2.4.2 Schnyder Orders . . . 53

2.4.3 Left-Right Edge Coloring 54
2.4.4 Experimental Evaluation 59
2.5 Stronger ILP Models Based on Small Cycles 71
2.5.1 Cycle Model 73
2.5.2 Strengthening the Cycle Model 77
2.5.3 Experiments 87

3 Genus **95**
3.1 Realizability Model 97
3.2 Small Faces 99
3.3 Additional Tuning 103
3.4 Experiments 105

4 Crossing Number **113**
4.1 Proof System 114
4.1.1 Known ILP Models 115
4.1.2 Design of Proof System 118
4.1.3 Modified-SECM 121
4.1.4 Verification Procedure 122
4.1.5 Practice and Experiments 124
4.2 Bounded Degree in Crossing Critical Graphs 126
4.2.1 13-Crossing-Critical Graphs with Large Degree 128
4.2.2 Arbitrary Criticality and Many Large-Degree Nodes . 135
4.3 Extending Simple Drawings 139
4.3.1 Reduction from T3-SAT to SSEI. 140

5 Conclusion and Outlook **147**

Bibliography **151**

List of Figures **172**

List of Tables **173**

Curriculum Vitae **175**

Chapter 1

Introduction

Cycle (*graph theory*). A closed walk in a graph that contains no repeated nodes.

Parts of this thesis were published in

- Markus Chimani and Tilo Wiedera. "An ILP-based Proof System for the Crossing Number Problem." In: *Proceedings of the 24th Annual European Symposium on Algorithms (ESA 2016), Aarhus, Denmark*. Vol. 57. LIPIcs, 2016, pp. 29:1–29:13. DOI: 10.4230/LIPIcs.ESA.2016.29
- Markus Chimani, Ivo Hedtke, and Tilo Wiedera. "Limits of Greedy Approximation Algorithms for the Maximum Planar Subgraph Problem." In: *Proceedings of the 27th International Workshop on Combinatorial Algorithms (IWOCA 2016), Helsinki, Finland*. Vol. 9843. LNCS, 2016, pp. 334–346. DOI: 10.1007/978-3-319-44543-4_26
- Markus Chimani, Karsten Klein, and Tilo Wiedera. "A Note on the Practicality of Maximal Planar Subgraph Algorithms." In: *Proceedings of the 24th International Symposium on Graph Drawing and Network Visualization, (GD 2016), Athens, Greece*. Vol. 9801. LNCS, 2016, pp. 357–364. DOI: 10.1007/978-3-319-50106-2_28
- Markus Chimani and Tilo Wiedera. "Cycles to the Rescue! Novel Constraints to Compute Maximum Planar Subgraphs Fast." In: *Proceedings of the 26th Annual European Symposium on Algorithms, (ESA 2018), Helsinki, Finland*. Vol. 112. LIPIcs, 2018, pp. 19:1–19:14. DOI: 10.4230/LIPIcs.ESA.2018.19

- Markus Chimani, Ivo Hedtke, and Tilo Wiedera. "Exact Algorithms for the Maximum Planar Subgraph Problem: New Models and Experiments." In: *ACM Journal of Experimental Algorithmics* 24.1 (2019). DOI: 10.1145/3320344. A preliminary version appeared in [CHW16].

- Markus Chimani and Tilo Wiedera. "Stronger ILPs for the Graph Genus Problem." In: *Proceedings of the 27th Annual European Symposium on Algorithms (ESA 2019), Munich/Garching, Germany*. Vol. 144. LIPIcs. 2019, pp. 30:1–30:15. DOI: 10.4230/LIPIcs.ESA.2019.30

- Drago Bokal, Zdeněk Dvořák, Petr Hliněný, Jesús Leaños, Bojan Mohar, and Tilo Wiedera. "Bounded Degree Conjecture Holds Precisely for c-Crossing-Critical Graphs with $c \leq 12$." In: *Proceedings of the 35th International Symposium on Computational Geometry (SoCG 2019), Portland, Oregon, USA*. vol. 129. LIPIcs. 2019, pp. 14:1–14:15. DOI: 10.4230/LIPIcs.SoCG.2019.14

- Alan Arroyo, Fabian Klute, Irene Parada, Raimund Seidel, Birgit Vogtenhuber, and Tilo Wiedera. "Inserting one edge into a simple drawing is hard." In: *Proceedings of the 46th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2020), Leeds, West Yorkshire, Great Britain*. LNCS. to appear. Springer, 2020

1.1 Basic Notation

Let us begin by defining the basic notation that we are going to use throughout this thesis. For a number $k \in \mathbb{N}$, we denote the set $\{1, 2, ..., k\}$ by $[k]$ and the set $\{0\} \cup [k]$ by $[\![k]\!]$. Given a set X, we refer to the set that contains all subsets of cardinality k of X as $X^{\{k\}}$ and to the set of all permutations with length k of elements from X as $X^{(k)}$. We may write $X \uplus Y$ instead of $X \cup Y$ to indicate that the set X is disjoint from the set Y. A *partition* of X is a set P of *classes* (or *partition sets*), such that $\biguplus_{p \in P} p = X$.

1.2 Complexity Theory

Here, we give a brief overview of those core aspects of algorithmic complexity theory that we are going to use throughout this thesis. Consider a finite set Σ that we call the *alphabet*. We refer to finite sequences of elements of Σ as *words*. The set of all words is denoted by Σ^*. Formally, a *decision problem* is a (usually infinite) set $\mathcal{P} \subseteq \Sigma^*$ of words. For such a prescribed set $\mathcal{P}$,

Table 1.1: Overview of common identifiers.

G, H	graph
n	number of nodes
m	number of edges
u, v, w	node
a, e, f	arc / edge
p	path / walk
c	closed walk / cycle
U, V, W	set of nodes
A, E, F	set of arcs or edges
$w(\cdot)$	(integral) weight of edges
D	drawing
Π	embedding
$skew$	skewness
gen	genus
cr	crossing number
ϕ	girth
x, y, z	variable in ILP or PBS
OPT (HEU)	optimal (resp. heuristic) solution value
α, β, γ	approximation factors

the respective problem is then to decide for any given word w, i.e., *problem instance*, whether $w \in \mathcal{P}$. We are typically interested in the resource demand for any such decision and its dependence on the length of w. Commonly considered resources are time and space.

Complexity class P contains exactly those decision problems that are solvable in deterministic polynomial time, i.e., problems that admit a polynomial function $f\colon \mathbb{N} \to \mathbb{N}$ such that solving the respective problem for a word of length n takes at most $f(n)$ steps of a deterministic algorithm. Complexity class NP, on the other hand, contains exactly the decision problems that are solvable in *non-deterministic* polynomial time. Equivalently, NP is the set of problems where for each word $w \in \mathcal{P}$ a *witness* w', i.e., another word w' exists, such that we can verify in deterministic polynomial time that $w \in \mathcal{P}$ by inspecting w'.

The above definitions are obviously incomplete as we did not formally establish the meaning of (non-)determinism nor what exactly constitutes a step in a deterministic algorithm. However, these considerations would stray too far from the actual scope of this thesis and we refer the interested reader to [HS11] for a detailed presentation of the topic. Still, to give some intuition on the topic, we may think of deterministic algorithms as algorithms that can be executed by ordinary real-world computers while omitting any memory limitations. The atomic steps of such algorithms are operations that take constant time on a given machine and we may think of them as central processing unit (CPU) instructions. From now on, we will only consider deterministic algorithms.

Clearly, P $\subseteq$ NP. One of the central open questions in computational complexity is, whether P is in fact the same as NP. Essentially, the question is whether all problems in NP can be solved in polynomial time. There are many problems in NP where all known algorithms for these problems require an exponential number of steps. Let us recapitulate the well-established concept of reductions that offers a way to distinguish these problems from those known to be in P.

A *reduction* from a decision problem A to another decision problem B is a function $f\colon \Sigma^* \to \Sigma^*$ such that $w \in A \iff f(w) \in B$. Assuming that evaluating the function requires only polynomially many steps and problem B is solvable in polynomial time as well, we obtain a polynomial time algorithm for problem A, i.e., problem B is at least as hard to solve as problem A. Given a problem B, we say that B is NP-*hard* if for each problem A in NP there is a reduction from A to B in polynomial time. Evidently, this property is transitive and allows us to characterize the hardest problems in NP, thus delimiting them from the ones that we know to be contained in P.

Optimization problems are different from decision problems in that they ask to maximize (or minimize) a quantity for a given word. However, both problem types are strongly related, as one may turn any optimization problem into a decision problem by asking whether a prescribed value can be achieved (and encoding this question in the word). As such, we may also classify optimization problems as NP-hard if they have a respective decision problem that is NP-hard.

Besides solving optimization problems exactly, one often aims to quickly find a solution that is provably close to the optimal solution value. Such algorithms are called *approximations*. An α-approximation or an approximation algorithm with ratio (or factor) α achieves a solution whose value HEU is close to the optimal value OPT in the following sense: For maximization problems, we have $HEU/OPT \geq \alpha$. Conversely, for minimization problems, we have $HEU/OPT \leq \alpha$.

Besides the complexity classes P and NP there are many more and hardness for these classes can be defined in similar ways. In this thesis we will, for example, consider the class MaxSNP that contains all optimization problems that allow constant-factor approximations in polynomial time. The hardest problems w.r.t. this class, i.e., the MaxSNP-hard problems, each have a constant bound $\neq 1$ that limits the achievable approximation ratio.

While problems are formally defined on words, from now on, we are going to consider graphs instead of words. They can be en- and decoded as words in linear time and offer a far more intuitive understanding of the problems at hand.

1.3 Graph Theory

Let us define the essential terms that graph theory is based upon. We aim to respect the terminology and definitions of Diestel's standard textbook on this topic and refer the interested reader to the book if she should desire a broader introduction [Die12].

Definition 1.3.1 (Simple Undirected Graph)**.** *Given a finite set V, a* simple undirected graph $G = (V, E)$ *on V is a 2-tuple, consisting of the set V and an irreflexive, symmetric binary relation $E \subseteq V^{\{2\}}$ on it. The elements of V are called the* nodes *or* vertices *of G. Elements of E are referred to as* edges*.*

For a given graph G we denote its node (edge) set by $V(G)$ (resp. $E(G)$). Edges are cardinality-2 subsets of the node set. If there is no ambiguity, we may abbreviate edge $\{v, w\}$ as vw or—equivalently—as wv. A node v

is called *adjacent* to another node w if $vw \in E$, in this case, v is referred to as a *neighbor* of w. Analogously, an edge e is said to be *adjacent* to another edge f if they share a common node, i.e., $e \cap f \neq \varnothing$. For a node v and an edge e, such that $v \in e$, we say that v is *incident* to e, and vice versa. A pair of non-adjacent nodes (or edges) is called *independent*. Given a node v, we denote its neighbors by $\mathrm{N}(v)$. Similarly, its set of incident edges is referred to as $\delta(v)$ and we generalize this notion to arbitrary subsets of the node set: For $W \subseteq V(G)$, let $\delta(W) \coloneqq \{e \in E(G) : |e \cap W| = 1\}$, this is also referred to as the *W-induced cut*. The *degree* $\deg(v)$ (occasionally, also referred to as *valency*) of a node v is simply its number $|\delta(v)|$ of incident edges. A node v with $\deg(v) = 0$ ($\deg(v) = 1$) is called an *isolated* node (*leaf*, respectively). Typically, we will denote the number $|V(G)|$ of nodes by n and the number $|E(G)|$ of edges by m. A graph on n nodes is a graph of *order* n.

Two graphs G, G' are called *isomorphic* to each other if there exists a bijection, $f \colon V(G) \to V(G')$, such that $\{u, v\} \in E(G) \iff \{f(u), f(v)\} \in E(G')$. We will usually consider isomorphic graphs as the same graph, specifying only the cardinality of $V(G)$ while disregarding the exact nature of its elements.

We say that a graph G is *complete* if $E(G) = V(G)^{\{2\}}$, i.e., there is no graph of the same order that contains more edges. The complete graph on n nodes is denoted as K_n, if it is part of a larger graph (subgraph, see below) it is also called an *n-clique*. If there exists a subset $W \subseteq V(G)$ such that $\delta(W) = E(G)$, we say that G is *bipartite*, i.e., each of its edges connects a node of W to a node not in W. A graph G that satisfies $E(G) = \{vw : v \in V(G) \setminus W \wedge w \in W\}$ is bipartite and has the maximum number of edges among all bipartite graphs of that order. As such, it is referred to as a *complete bipartite* graph (albeit not being complete). For $n_1 \coloneqq |W|$ and $n_2 \coloneqq |V(G) \setminus W|$, we denote the respective complete bipartite graph by K_{n_1,n_2}. A graph where each node has the same degree d is called a *d-regular* graph. If the graph is 3-regular, it is also referred to as a *cubic* graph.

Subgraphs and Walks. Given a graph G, we may consider another graph H that is contained in G, i.e., $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. In this case, we say that H is a subgraph of G. For a node subset $W \subseteq V(G)$ we call the graph $(W, \{e \in E(G) : e \subseteq W\})$, the W-induced subgraph of G and refer to it by $G[W]$. It contains exactly those edges of G that are only incident with W. Given an edge subset $F \subseteq E(G)$, the F-induced subgraph $G[F]$

of G is defined as $(\bigcup_{e\in F} e, F)$. Analogously, it contains exactly those nodes of G that are incident with an edge in F. We say that a subgraph H of G is *spanning*, if $V(H) = V(G)$. When considering multiple graphs on a common node set, we may write $\delta_H(v)$ ($\mathrm{N}_H(v)$) to indicate that we refer to $\delta(v)$ ($\mathrm{N}(v)$, respectively) w.r.t. graph H.

A *walk* $p = (e_1, e_2, ..., e_k)$ is a sequence of edges, such that for all $i \in [k-1]$ we have $|e_i \cap e_{i+1}| = 1$ and for all $i \in [k-2]$ we have $e_i \cap e_{i+1} \neq e_{i+1} \cap e_{i+2}$. A walk that is a k-tuple is a walk of *length* k, or a k-walk. For $k \geq 2$, we refer to the single node of $e_1 \setminus e_2$ as the *start* and to the single node $e_k \setminus e_{k-1}$ as the *end* of p. Both nodes are called the *endpoints* of p. For $k = 1$, the endpoints are the nodes incident with the only edge. A walk with start u and end v, is a walk from u to v, or a u-v-walk. A walk that starts and ends at the same node, is a *cyclic* or *closed* walk. In the degenerate case that $k = 0$, we say that p is the *empty* walk and may call it a (cyclic) v-v-walk for any node v. If the edges of a walk p are pairwise different, we say that p is *internally edge-disjoint*. Similarly, if $e_i \cap e_j = \varnothing$ for all $i, j \in [k]$ where $|i - j| > 1$ the walk is called an *internally node-disjoint* walk, a *simple* walk, or—even shorter—a *path*. Whenever convenient, we may consider any walk $p = (e_1, e_2, ..., e_k)$ in a graph G as the respective subgraph $G[\bigcup_{e\in[k]} e_k]$ of G (instead of p being a sequence).

Given two nodes u, v and a number $t \in \mathbb{N}$, we refer to a collection of t internally disjoint u-v-paths, each of length two, as a u-v-bundle of thickness t, or $\mathcal{B}^t_{u,v}$ in short. Given a bundle $\mathcal{B}^t_{u,v}$, we denote its t inner nodes of degree two, i.e., $V(\mathcal{B}^t_{u,v}) \setminus \{u, v\}$, by $\mathcal{I}(\mathcal{B}^t_{u,v})$.

Connectivity and Cycles. We say that a pair $v, w \in V(G)$ of nodes is *connected* (w.r.t. a graph G) if there is a v-w-path (in G). A graph G is *connected* if every pair of its nodes is connected. Otherwise, we say that G is *disconnected*. A connected 2-regular graph is called a *cycle*. Any cycle is also a cyclic walk and the previous notations for general walks apply. Given a cycle c, any edge not in $E(c)$ that is adjacent to two nodes of c, is called a *chord* of c. A graph that does not contain any cycles is referred to as a *forest*, or an *acyclic* graph. A connected forest is called a *tree*. If a shortest cycle of a graph G has length ϕ, then ϕ is referred to as the *girth* of G.

Consider a graph G and a partition $W_1, W_2, ..., W_k$ of its node set such that any two nodes u, v are connected if and only if they are in the same partition set, i.e., for each pair u, v of nodes, there is an $i \in [k]$ with $\{u, v\} \subseteq W_i$ if and only if G contains a u-v-path. Then, $G_1, G_2, ..., G_k$ where $G_i \coloneqq G[W_i]$ are called the (connected) *components* of G. More generally, we

may define *ℓ-connected graphs* (*ℓ-edge-connected graphs*) as those that stay connected when removing at most $\ell - 1$ nodes (edges, respectively). The maximal ℓ-connected subgraphs of a graph are its *ℓ-connected components*, these more general components are no longer pairwise node-disjoint. In particular, we will consider the case $\ell = 2$, i.e., *biconnected components* that are also referred to *bicomps* and *blocks*. A block that is a single edge is a *bridge*.

Operations on Graphs. Let us introduce some shorthand notations for modifying a given graph G: The addition of a new node v (new edge e) is denoted by $G \oplus v \coloneqq (V(G) \uplus \{v\}, E(G))$ (and $G \boxplus e \coloneqq (V(G), E(G) \uplus \{e\})$, respectively). Similarly, we abbreviate the removal of an existing node $v \in V(G)$ (edge $e \in E(G)$) by $G \ominus v \coloneqq G[V(G) \setminus \{v\}]$ (and $G \boxminus e \coloneqq (V(G), E(G) \setminus \{e\})$, respectively). Consider an edge $vw \in E(G)$. Then, the operation that turns G into $G \oplus v' \boxplus vv' \boxplus v'w \boxminus vw$ is called an *edge split* of edge vw. It transforms vw into a 2-path and introduces a new node v'. If a graph can be obtained by a sequence of edge splits from G, it is called a *subdivision* of G. An *edge contraction* of an edge vw identifies its incident nodes, i.e., it transforms G into the new graph $(V(G) \setminus \{w\}, \{e \in E(G) : w \notin e\} \cup \{uv : u \in \mathrm{N}(w) \setminus \{v\}\}$. A graph that is obtained by a sequence of edge contractions and edge deletions from G, is referred to as a *minor* of G. Consider two graphs G, G' that may have some nodes in common. We denote the *union* or *join* of these graphs by $G \sqcup G' \coloneqq (V(G) \cup V(G'), E(G) \cup E(G'))$.

Directed Graphs. A common generalization of undirected graphs is to consider directed edges, usually called *arcs*. This is equivalent to dropping the requirement for symmetry on the graph's binary relation:

Definition 1.3.2 (Simple Directed Graph)**.** *Given a finite set V, a* simple directed graph *$G = (V, A)$ on V is a 2-tuple, consisting of the set V and an irreflexive, binary relation $A \subseteq V^2$ on it. The elements of V are called the* nodes *or* vertices *of G. Elements of A are referred to as* arcs*.*

We straightforwardly use the same notation as for undirected graphs that we amend as follows: Given a directed graph G we refer to its arc set as $A(G)$. Given an arc $a = vw$, i.e., an directed edge from v to w, we denote its undirected counterpart $\{v, w\}$ by $\mathrm{e}(a)$ and its reversal wv as $\mathrm{rev}(a)$. Given a node-subset W, we refer to its incoming (outgoing) arcs as $\delta^-(W)$ ($\delta^+(W)$, respectively). Here, we may omit curly braces when referring to node-sets of cardinality 1. Given two node-subsets W, W' and a set A of

arcs, we refer to the set of all arcs in A that go from W to W' as $W \times_A W'$. Consider a directed graph G and the undirected graph $G' := (V(G), E)$, where $E := \{\mathrm{e}(a) \mid a \in A(G)\}$. We say that G' is the *shadow* of G. If $|E(G')| = |A(G)|$, G is an *orientation* of G'.

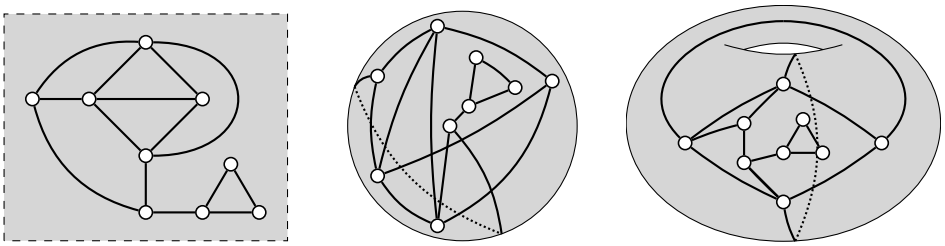
Multigraphs. In some cases, e.g., for dual graphs (defined in the next section) we need to consider graphs where there is more than one (undirected) edge between a pair of nodes. We say that such edges are *parallel* to each other. Formally, this concept does not fit our definition of graphs but it is easily defined by replacing the set of edges/arcs by a multiset. Similarly, we may consider graphs with *self-loops*, i.e., edges/arcs that are incident with exactly one node (twice). Considering our previous definition of graphs, this is achieved by dropping the requirement of irreflexivity. We refer to these graphs with self-loops and/or parallel edges as *multigraphs* and aim to use the same notation for them as for simple graphs (while being extra careful to avoid ambiguity). Given an undirected graph G, we may consider a directed graph H with $2|E(G)| = |A(H)|$ and such that for each edge $\{u, v\}$ of G both arcs uv and vu are contained in $A(H)$. In this case, A is called the *bidirected* graph of G.

1.4 Planarity

Graphs are typically visualized by lines (representing edges) and points (representing nodes). Intuitively, planar graphs are those graphs that can be drawn on the plane without any edges intersecting each other. Apart from their merits for visual representation, these graphs play an important role in algorithmic complexity and are a corner stone of graph theory. Despite this seemingly simple notion of planarity, we need some rigorous definitions to obtain a formal description of planar graphs. To this end, we follow the terminology of the excellent textbook by Mohar and Thomassen [MT01] on embedded graphs.

Definition 1.4.1 (Surface)**.** *A* surface S *is a compact, connected,* 2*-dimensional manifold. The* genus *of* S *is the maximum number of non-intersecting closed Jordan curves on* S *whose removal does not render* S *disconnected.*

Typically, we will consider orientable surfaces and often in its simplest form, i.e., the Euclidean plane $\mathbb{R}^2$ or—almost equivalently—the (surface $\mathcal{S}^2$ of a) sphere.



(a) planar on $\mathbb{R}^2$ (b) non-planar on $\mathcal{S}^2$ (c) planar on the torus

Figure 1.1: Different drawings of the same graph. Dotted lines run on those parts of the surface (gray) that are not facing the reader.

Definition 1.4.2 (Drawing)**.** *A* drawing *of a graph G on a surface S is a bijection f_V from $V(G)$ to a set of points on S, and another bijection f_E from $E(G)$ to a set of Jordan arcs on S, such that for each edge $vw \in E(G)$, the boundary points of $f_E(vw)$ are the points $f_V(v)$ and $f_V(w)$ and the interior of $f_E(vw)$ is disjoint from $f_V\big(V(G)\big)$.*

If the intersection of each pair of Jordan arcs contains at most one point, we say that the drawing is *simple*. Intuitively, a drawing is a visualization of a graph where edges are not allowed to pass through nodes. See Fig. 1.1 for an example of different (simple) drawings of the same graph.

Definition 1.4.3 (Planar Drawing/Embedding)**.** *A drawing is* planar *if and only if its arcs' interiors are pairwise disjoint. This is also referred to as an* embedding*.*

Definition 1.4.4 (Planar Graph)**.** *A graph is* planar *if and only if it admits an embedding on the plane $\mathbb{R}^2$ (or—equivalently—on the sphere $\mathbb{S}^2$).*

Arguably, this definition is a bit awkward as it requires several concepts that are rather close to the actual process of physically drawing a graph. On the other hand, it obviously grasps the intuitive notion of planarity. We will later see more abstract and combinatorial characterizations of planar graphs.

Usually, we are not concerned with all possible drawings of a given graph. Instead, we divide the set of drawings into equivalence classes, where drawings are represented by their rotation systems:

Definition 1.4.5 (Rotation System)**.** *Any drawing of a graph on an orientable surface, induces a (clockwise) cyclic order of incident edges around each of its nodes. This set of cyclic orders, or* rotations*, forms a* rotation system*.*

As one can imagine, rotation systems allow much simpler representations of drawings—a handy fact when designing algorithms. Still, rotation systems usually suffice to capture all relevant properties of their drawing (or rather *drawings*). To give some intuition on the nature of rotation systems, consider the following fact that we do not prove here: A homeomorphic transformation of a drawing maintains its rotation system (up to mirroring) [MT01]. In particular, each planar drawing is—up to homeomorphic transformation—fully specified by its respective rotation system. See Fig. 1.2a for an example of a planar drawing and its rotation system.

Definition 1.4.6 (Face)**.** *Consider an embedding* Π *of a graph on a surface* S *and let* J *denote its set of Jordan arcs. By the Jordan Curve Theorem [MT01, Section 2.1] and its generalization, we obtain the following: Removing* J *from* S *partitions the previously connected surface into a set of connected surfaces. These new surfaces are called the* faces *of* Π*.*

Observe that the border of each face corresponds to a closed walk in the graph. Similar to our previous definitions, if an edge e (node v) is part of the border of a face f, then f is *incident* with e (v, respectively) and vice-versa. If a (planar) graph G has an embedding in the plane such that there is a face that is incident with each node of G, then G is referred to as an *outerplanar* graph.

Already in 1758, Euler found a formula to characterize topological spaces that gives rise to a surprising relation on embedded graphs:

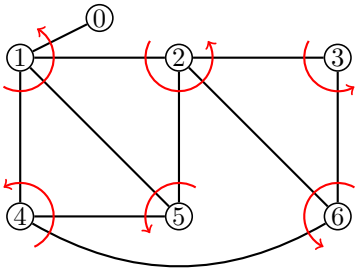
Theorem 1.4.7 (Euler's Formula)**.** *Consider an embedding* Π *of a graph* G *on an orientable surface* S *of genus* gen*. We denote the number of faces of* Π *by* f*. It holds that* $|V(G)| - |E(G)| + f \geq 2 - gen$ *and the relation is satisfied with equality if* Π *is a genus-minimal embedding of its rotation system.*

The consideration of embeddings provides a surprising duality between any embedded graph and its respectively embedded "dual". To formally define this, we require the previous concept of multigraphs.

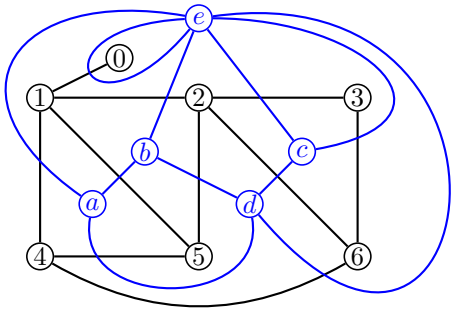
Definition 1.4.8 (Dual Graph)**.** *Given an embedding* Π *of a* (primal) *graph* G*, the* dual graph G' *of* Π *is a multigraph that has the faces of* Π *as its nodes and there is a bijection from* $E(G)$ *to* $E(G')$ *such that each primal edge* $e \in E(G)$ *corresponds to a dual edge* $\in E(G')$ *connecting the face on the left of* e *with that on the right of* e*.*

See Fig. 1.2b for an example of a planar drawing and its dual.

Given any embedding, it is straightforward to obtain an embedding of its dual graph on the same surface: Place each dual node on its primal



(a) A rotation system (red) of a planar drawing in the plane. Here, the rotation at node 1 is $\langle 4, 5, 2, 0\rangle$ which is equivalent to $\langle 5, 2, 0, 4\rangle$, $\langle 2, 0, 4, 5\rangle$, and $\langle 0, 4, 5, 2\rangle$ by cyclicity. On the other hand, node 3 has degree only two and hence admits exactly one cyclic rotation, independent of the drawing.



(b) The dual graph (blue). It contains parallel edges ce and a self-loop at node e. Note that the dual is again planar and each crossing occurs between a dual (blue) edge and its primal (black) counterpart. Conversely, the dual of the dual graph (blue) is the primal graph (black).

Figure 1.2: Rotation system (a) and dual graph (b) of the same planar drawing.

connected surface and for each dual edge, cross its corresponding primal Jordan arc. The rotation system of this new embedding directly corresponds to the cyclic order of edges along each primal face's boundary.

Despite the previous definition of planar graphs that appears quite intuitive, one may hope for a more combinatorial perspective on the matter. The first such characterization of planar graphs was achieved by Kuratowski in 1930:

Definition 1.4.9 (Kuratowski Subdivision)**.** *Any subdivision K of the K_5 or the $K_{3,3}$ is called a* Kuratowski subdivision*. Those nodes that each have degree at least* 3 *in K are the* Kuratowski nodes *of K.*

Theorem 1.4.10 (Kuratowski's Theorem [Kur30])**.** *A graph is planar if and only if it does not contain a Kuratowski subdivision as a subgraph.*

Shortly after, Wagner obtained a similar result that employs a similar but slightly different idea.

Theorem 1.4.11 (Wagner's Theorem [Wag37])**.** *A graph is planar if and only if neither contains the $K_{3,3}$ nor the K_5 as a minor.*

In contrast to Kuratowski, Wagner uses the concept of graph minors. It later turned out that this newly coined concept gives rise to fundamental and deep results in graph theory. Particularly, embeddability on any given orientable or non-orientable surfaces with fixed genus is characterized by the appearance of (at least one of) a *finite* set of forbidden minors [RS90]. In fact, for each surface S, there is a polynomial time algorithm that tests embeddability on S [RS04; RS95]. In fact, there is a linear time algorithm for orientable surfaces [Moh99]

1.5 Measures of Non-Planarity

Many algorithmic problems that currently require a staggering amount of time to be solved in general, can be solved faster when restricted to planar graphs. To name a prominent example, the planar separator theorem asserts that for any planar graph G of order n, there is a set $W \subseteq V(G)$ of $\mathcal{O}(\sqrt{n})$ nodes such that each component of $G[V \setminus W]$ has at most $2n/3$ nodes [LT80], essentially providing an operation to recursively split suitable problems on planar graphs into smaller (nearly) independent ones. This result gave rise to a broad set of divide and conquer algorithms, including a linear time algorithm for single source shortest path with non-negative edge weight. In fact, the same algorithm can also be used to compute, in linear time, a maximum s-t-flow if s and t are incident to a common face [Hen+97]. It is long known that planar graph isomorphism is polynomial-time solvable [HT71]. There are constant-factor approximations for maximum independent set [Bak94], traveling salesperson [Kle08], and even a PTAS for Steiner tree [BKK07] on planar graphs. The latter two problems also allow exact subexponential time algorithms [MPP18]. It is evident that one may consider planar graphs to be—in some sense—better than their general counterparts as they allow for faster algorithms (and, evidently, nicer representations).

On the other hand, even non-planar graphs are not uniformly bad w.r.t. these considerations: there are non-planar graphs that allow good visualizations and even broad classes of non-planar graphs where many problems can be solved faster than for the general case. Often, these graphs exhibit similar properties as planar ones. The probably richest such class contains the graphs of bounded genus (defined later, cf. Section 1.5.4) and exploring its algorithmic properties is a vivid research field that spawned many results. Graph genus plays a key role for the complexity of certain problems, in particular w.r.t. polynomial-time approximation scheme (PTAS) and fixed parameter tractability (FPT) [Che+07]. Algorithms tailored to achieve faster

running time on planar graphs can often be adapted to the bounded genus setting. For example, a bounded graph genus leads to: linear-time graph isomorphism testing [Che94]; FPT running time for dominating set [EFF04]; subexponential FPT running time for many bidimensional problems, including vertex cover and variants of dominating set [DHT06]; a quasi-PTAS for capacitated vehicle routing [BKS17]; stronger preprocessing for several Steiner problems [MPP18]; and many more. Other measures of non-planarity are often lower bounded by the genus, and similar results for these (more restrictive) measures are rather scarce: Typically, the algorithm for bounded genus cannot be improved there. To name a few exceptions, on graphs with bounded skewness (also defined later, cf. Section 1.5.1) for example, there is a polynomial time algorithm for weighted maximum cut [Chi+20], a maximum flow algorithm that achieves the same running time as for the planar case [HW07], and an approximation algorithm for the crossing number [CH17].

It appears natural to ask "how far away from being planar" a given non-planar graph is. To answer this question, many measures have been proposed over the years and indeed there is no general measure suited for every task but the choice heavily depends on the application in mind. There are, however, some measures that are well established, each with a rich history of results. We will introduce and discuss the measures skewness (and maximum planar subgraph), vertex deletion number (and maximum planar induced subgraph), splitting number, genus, crossing number, thickness, and coarseness. We present this set of measures as most of its elements are thoroughly studied (albeit many open question still remain) and sufficiently diverse: Most other measures of non-planarity are—in one way or another—derived from them. In fact, most of these measures have several variations that have also been studied intensively. A very prominent example is the abundance of crossing number variants [Sch18].

For some measures we will also define an edge-weighted variant since standard preprocessing routines transform large unweighted graphs to smaller weighted ones [CG09].
In contrast to a survey from 2001 by Liebers [Lie01], we focus only on algorithmic results and highlight some recent advances. Particularly, we are going to omit any results on bounds or even exact formulae on specific graph classes. For example, such results are often obtained for complete (bipartite) graphs.

Apart from their merits for visualizations and faster computations, measures of non-planarity are also of immediate practical interest. For example, already in 1976 the problem of determining the (weighted) skewness of a

graph was considered to be a core difficulty in designing efficient industrial facility layouts [FGG85; FR76]. Modern chip design, i.e., very large-scale integration (VLSI), requires to print huge integrated circuits. Since these circuits are printed on a plane surface (the *die*) only planar circuits can be printed directly and non-planar ones require a way to circumvent short circuits. This can, for example, be achieved by replacing each crossing with a new wire (crossing number), replacing entire edges (skewness), printing the circuit on multiple, interconnected dies (thickness), or adding new wires between certain nodes (splitting number).

1.5.1 Skewness and Maximum Planar Subgraph

When considering the skewness, we are measuring non-planarity by the number of obstructing edges. More precisely, skewness is defined as follows:

Definition 1.5.1 (Skewness)**.** *The* skewness *skew*(G) *of a graph* G *is the minimum cardinality of an edge set* F *whose removal turns* G *into a planar graph, i.e.,* $G[E(G) \setminus F]$ *is planar (we say that removing* F planarizes G*). Given a weight function* w *on the edges, we define skew*(G) *as* $\min \sum_{e \in F} w(e)$ *such that removing* F *planarizes* G*.*

Occasionally, skewness is also referred to as *edge deletion number* (borrowing from the term "vertex deletion number", to be defined below). The respective decision problem is also known as *non-planar deletion*. A subgraph that achieves this value is called a *maximum planar subgraph*. In contrast, a *maximal* planar subgraph of a graph G is one that does not admit adding any new edges from G while maintaining planarity.

The problem of determining *skew* is known to be NP-hard [LG79; WAN83; Yan78] and was mentioned already in the classical textbook by Garey and Johnson [GJ79]. In fact, a straightforward reduction from Hamiltonian cycle (HC) exists [CHW16]. The problem remains NP-hard even for cubic (and hence also K_5-subdivision free) graphs [FFM01].

Both, skewness as well as maximum planar subgraph (MPS) are MaxSNP-hard [Căl+98]. It was later shown that the latter problem remains MaxSNP-hard on cubic graphs [FFM04]. A constant factor 4/9-approximation algorithm for maximum planar subgraph exists [Căl+98]. The algorithm is based on certain, almost acyclic graphs, called *cacti*. Surprisingly, as of today, and despite various attempts [CHW16; Cim95a; CS17; Por08], no better approximation algorithm has been identified and no constant factor approximation of skewness is known.

The first—and to date (except for our new results herein) only—non-trivial, exact algorithm is based on integer linear programs (ILPs) as well as Kuratowski's famous characterization of planarity. It was presented a quarter of a century ago by Mutzel [Mut94]. We will later build upon this result to design practically faster algorithms for skewness, cf. Section 2.5

1.5.2 Vertex Deletion Number and Maximum Planar Induced Subgraph

Instead of removing edges, we may also remove nodes from a graph to obtain a smaller but planar graph:

Definition 1.5.2 (Vertex Deletion Number)**.** *The* vertex deletion number *vdel*(G) *of a graph* G *is the minimum cardinality of a node subset* $W \subseteq V$ *such that* $G[V \setminus W]$ *is planar.*

The problem of deciding the vertex deletion number is also referred to as the *k-apex problem* and *non-planar vertex deletion.* A graph that achieves this value is called a *maximum induced planar subgraph.* By an L-reduction (i.e., a special form of approximation-preserving reduction) from skewness, vertex deletion number is MaxSNP-hard on cubic graphs [Far+06]. The inverse measure that maximizes the order over all planar induced subgraphs, i.e., maximum induced planar subgraph (MIPS), does not allow a constant-factor approximation, unless $\mathrm{P} = \mathrm{NP}$, by an approximation preserving reduction from independent set [Far+06]. For graphs with bounded maximum degree Δ, however, approximation algorithms do exist, e.g., a 3/4-approximation on cubic graphs and a $3/(\Delta + 5/3)$ on general such graphs [Far+04; MF07]. In fact, there always is a large MIPS if the *average* degree is bounded and respective approximations are easily derived [BEZ15; EF01; EF08]. Even an exact algorithm that achieves a surprisingly low theoretical running time was found [FTV11]. Generally, the problem appears closely related to the maximum independent set (MIS) problem. Since for each number k, the class of graphs with vertex deletion number k is closed under minor operations, it follows by deep results of Robertson and Seymour that the problem is in FPT w.r.t. its natural parameterization [RS04; RS95].

It was later shown that there also is a practically feasible FPT-algorithm that achieves quadratic running time for fixed vertex deletion number [MS12].

Since each planar graph on n nodes contains an independent set with cardinality at least $2n/9$ [Alb76], and each independent set is planar, MIPS and MIS behave similarly w.r.t. approximation: Any α-approximation for MIS implies a $2/9 \cdot \alpha$-approximation for MIPS and, conversely, any β-approximation

for MIPS implies a $2/9 \cdot \beta$-approximation for MIS. This, e.g., shows that MIPS is constant-factor approximable on graphs of bounded genus [GHT84].

1.5.3 Splitting Number

Instead of removing an obstructing node, we may also split it into multiple nodes, each with a smaller neighborhood.

Consider the following operation: Pick two nodes u, v in a graph and identify them. Clearly, this operation may turn a planar graph into a non-planar one, e.g., we may iteratively turn a matching, consisting of 10 independent edges, into a K_5. The splitting number considers the inverse of this operation: A *split operation* s on a given graph G replaces a node w by two new nodes u, v such that $\mathrm{N}(u) \uplus \mathrm{N}(v) = \mathrm{N}(w)$. We denote the graph resulting from the application of s to G by $s(G)$.

Definition 1.5.3 (Splitting Number)**.** *The* splitting number *vsplit*(G) *of a graph* G *is the minimum length* k *of a sequence* $(s_1, ..., s_k)$ *of split operations* s_i*, such that* $s_k\big(...s_2(s_1(G))\big)$ *is planar.*

From the inverse perspective, we may also define the splitting number of G as the minimum number of node identifications over all planar graphs G' where we start with G' and produce G by iteratively identifying two nodes each.

Determining the splitting number is NP-hard by a reduction from 3-SAT [FFM01]. In fact, the problem is MaxSNP-hard already on cubic graphs [FFM04]. As of today, no constant factor approximation for splitting number is known.

1.5.4 Genus

Instead of modifying the graph until it becomes planar, we may also consider a broader class of drawings. Here, we consider embeddings on surfaces of higher genera.

Definition 1.5.4 (Genus)**.** *The (orientable)* genus *gen*(G) *of a graph* G *is the smallest number* k *such that* G *can be embedded on an orientable surface of genus* k*.*

Considering the relative popularity of this problem, its NP-hardness was established relatively late in 1989 [Tho89]. Shortly after, Thomassen published an algorithm that computes the genus of a restricted class of graphs in polynomial time [Tho90] (graphs that have so-called LEW-embeddings).

Once again, by seminal results of Robertson and Seymour (just as for vertex deletion number) the problem is FPT w.r.t. its natural parameterization [RS04; RS90; RS95]. However, their proposed algorithm is doubly exponential in $gen(G)$ and already for toroidal graphs it is highly infeasible in practice. The algorithm is based around the observation that for each fixed genus, there is a *finite* list of forbidden minors. It then suffices to test whether any such forbidden minor is present in G to decide whether G can be embedded in a given surface. Already for toroidal graphs, this list contains tens of thousands graphs. Mohar was the first to write down an explicit and constructive proof of such an algorithm and the respective series of papers spans more than a hundred pages [Moh99]. The algorithm was later simplified and extended to graphs of bounded treewidth [KMR08]. It remains open to find a practically feasible FPT algorithm [MK11]. Indeed, there is little algorithmic progress for this measure. This is particularly true w.r.t. practical algorithms: The first (non-trivial) general exact algorithm was described recently and is only able to solve toroidal instances in practice [Bey+16]. While some graph classes allow approximations of genus (or closely related measures), general such algorithms are not known [CS18; KS15; MNS17]. In fact, not even good heuristics for the genus are known. We will present new and faster algorithms for computing the genera of general graphs in Chapter 3 [CW19].

A similar notion for non-orientable surfaces, i.e., the non-orientable genus, and combinations of these measures, e.g., the Euler genus, have also been studied.

1.5.5 Crossing Number

Another way of generalizing the considered drawings is obtained by drawings in the plane that are not planar, in the sense that edges are allowed to cross each other. This is arguably the most natural way to draw a non-planar graph as the graph is visualized very similarly to planar ones and—as opposed to genus—we are accustomed to drawing on a plane surface, e.g., a simple sheet of paper.

Definition 1.5.5 (Crossing Number)**.** *The* crossing number *of a simple drawing D is the number of edge pairs that intersect each other. The* crossing number *$cr(G)$ of a graph G is the minimum crossing number over all simple drawings of G in $\mathbb{R}^2$.*

Crossing numbers have an extremely rich history since their initiation over 70 years ago. This is witnessed, for example, by Vrt'o's excellent bibliography

of over 700 publications as well as Schaefer's great book and online survey that list about 50 different crossing number variants [Sch13; Sch18; Vrt14].

Deciding the crossing number of general graphs is NP-hard [GJ79] and remains so even for highly restricted classes, such as cubic or near-planar graphs [CM13; Hli06]. The latter are those graphs that can be turned planar by removing a single edge. A PTAS is not possible unless P = NP and it remains open whether constant-factor polynomial time approximation algorithms do exist [Cab13]. On the positive side, the problem allows practically strong heuristics and restricted approximations [CG12; CH17; CHN19; GM03]. It is in FPT w.r.t. its natural parameterization and even w.r.t. skewness [KR07]. However, the respective algorithms are highly infeasible in practice. Particularly, the former algorithm's running time is doubly exponential in $cr(G)$.

The only known exact practical algorithms are based on ILPs [Buc+08; CMB08]. Unfortunately, both underlying models are too large to be solved directly. Instead, they require intricate implementations that use highly sophisticated column generation techniques to obtain a practically feasible algorithm [CGM10; Chi08; CMB08].

1.5.6 Thickness

We may also consider partitioning the edge set of a non-planar graph G to obtain several planar graphs whose union is G.

Definition 1.5.6 (Thickness)**.** *The* thickness *$thick(G)$ of a graph G is the minimum cardinality k of a partition $E_1, E_2, ..., E_k$ of its edge set such that $G[E_i]$ is planar for each $i \in 1, ..., k$.*

Thickness was first studied by Tutte in 1963 [Tut63] and proven to be NP-hard 20 years later [Man83]. While this problem received significant attention from a mathematical perspective [MOS98; MP12], much less has been achieved w.r.t. algorithms. On the positive side, there is a straightforward 3-approximation by arboricity, i.e., the minimum number of trees that G can be partitioned into [GW92]. Despite some effort, no better way of approximation has been found [KY03]. However, several heuristic approaches were studied over the years [Cim95b; KR02; MPV01; Por05].

1.5.7 Coarseness

Coarseness is a rather different measure in that it aims to *maximize* a quantity of non-planar structures. Still, this measure is a positive integer

that is minimal, i.e., zero, if and only if the respective graph is planar. It originated during a flawed description of thickness by Erdős who made an inadvertent mistake and thus accidentally coined the concept of coarseness [Har69].

Definition 1.5.7 (Coarseness)**.** *The* coarseness *coarse*(G) *of a graph* G *is the maximum number of edge-disjoint Kuratowski subdivisions contained in* G*.*

Unfortunately, this measure did not receive much scientific attention. Similarly to thickness, algorithmic results are scarce.

Hardness. To our best knowledge—albeit relatively simple to see—the NP-hardness of the coarseness problem was never established in literature. As such, we provide a formal proof for it. To this end, we use an established problem that considers the existence of disjoint paths.

Definition 1.5.8 (edge-disjoint paths problem)**.** *Given a graph* G*, and a set of node-pairs* $F \subseteq V^{\{2\}}$*, the problem of deciding whether a set* P *of pairwise edge-disjoint paths in* G *exists such that for each* $\{u, v\} \in F$ *there exists a path with endpoints* u *and* v *in* P*, is called the* edge-disjoint paths problem (EDP)*.*

EDP remains NP-hard when restricted to planar graphs [MP93; Vyg95]. Surprisingly, the problem turned out NP-hard already for series-parallel graphs although it admits polynomial time algorithms on outerplanar graphs [NVZ01].

Theorem 1.5.9. *Computing coarseness is* NP*-hard.*

Proof. We perform a reduction from planar EDP. Let (G, P) denote an instance of this problem. We construct G' in the following manner: Starting with G, for each pair $\{u, v\} \in F$, we create a new $K_{3,3}^-$, i.e., a $K_{3,3}$ where we deleted one edge. Let u' and v' denote the nodes of degree 2 in this $K_{3,3}^-$. Finally, we add the edges uu' and vv'. We will refer to these two edges as the *cut-edges* of $\{u, v\}$.

We now claim that *coarse*$(G') \geq |F|$ if and only if (G, F) is a yes-instance.

$\Longleftarrow$ Assume there exists a solution of (G, F). For each pair of F, its solution-path together with its cut-edges and $K_{3,3}^-$ forms a $K_{3,3}$-subdivision. These subdivisions are edge-disjoint by construction of G'.

$\Longrightarrow$ We observe that removing one cut-edge of each node-pair of F planarizes G': Its biconnected components are either subgraphs of G,

single edges, or $K_{3,3}^-$s, all of which are planar. Hence, each Kuratowski-subdivision of G' requires at least one such edge (and we have $coarse(G) \leq |F|$).

Recall that the proper nodes of a Kuratowski subdivision K are those with degree > 2 in K. Since each $K_{3,3}^-$ is attached by a 2-cut to two nodes u and v of G (by its cut-edges, see construction above), it follows that at most one path of one Kuratowski-subdivision K crosses this cut and all its proper nodes either lie in $V(G)$ or in the $K_{3,3}^-$. In both cases, we find a path in $K \cap E(G)$ with endpoints u, v. If follows that there exist $coarse(G)$ edge-disjoint paths in G, joining each pair in F. □

It was later shown that the maximization variant of EDP, i.e., finding a maximum cardinality subset of F such that each pair in F is assigned an edge-disjoint path, is MaxSNP-hard on trees of rings [Erl01]. A *tree of rings* is obtained by starting with a cycle and iteratively adding new (disjoint) cycles by identification of a node in the existing structure with a node in the new cycle. Unfortunately, although trees of rings are planar, we cannot directly use this to conclude MaxSNP-hardness of the graph coarseness problem since the construction in the above proof is not approximation preserving.

1.6 Integer Linear Programming

Here, we briefly discuss a particularly successful general approach for solving (NP-hard) problems. We will use it as a central toolbox to obtain fast algorithms for different measures of non-planarity. For a more in-depth discussion of the topic we refer the interested reader to [Sch99].

A *linear program (LP)* consists of a cost vector $c \in \mathbb{Q}^d$ together with a set of linear inequalities, called constraints, that define a polyhedron P in $\mathbb{R}^d$. This set of linear inequalities is typically represented by a rational $k \times d$-matrix A and a vector $b \in \mathbb{Q}^k$, where the polyhedron consists of the points $x \in \mathbb{R}^d$ that satisfy $Ax \leq b$. It is well known that given A, b, and c as the input, we can determine in polynomial time a point $x \in P$ that maximizes the objective function $c^\intercal x$. Assuming $\mathrm{P} \neq \mathrm{NP}$, this is no longer true when restricting x to have integral components; the so-modified program is an *integer linear program (ILP)* and we refer to the respective optimization problem as the *ILP problem*. Conversely, the *LP relaxation* of an ILP is obtained by dropping the integrality constraints on the components of x. The optimal value of an LP relaxation is a dual bound on the ILP's objective; e.g., an upper bound for maximization problems.

A reduction from a general problem to the ILP problem is an *ILP model* (or simply, *model*). As there are several ways to model a given problem as an ILP, one aims for models that can be efficiently computed and yield small dimensions as well as strong dual bounds, to achieve good practical performance. This is crucial, as ILP solvers are based on a branch-and-bound (B&B) scheme that relies on iteratively solving LP relaxations to obtain dual bounds on the ILP's objective. When a model contains too many constraints, it is often sufficient to use only a reasonably sized constraint subset to achieve provably optimal solutions. This allows us to add constraints during the solving process, which is called *separation*. We say that model A is *at least as strong* as model B, if for all instances, the LP relaxation's value of model A is no further from the ILP optimum than that of B. If there also exists an instance for which A's LP relaxation yields a tighter bound than that of B, then A is *stronger* than B.

Note that the scope of this thesis is not to evaluate the different strategies to solve general ILPs quickly. Instead, and among other considerations, we will gauge the theoretical and practical fitness of different ILP models for the same problems. To this end, we assume some established theoretical facts on the solving process and use state-of-the-art implementations of the fastest solvers for our practical evaluations.

Pseudo-Boolean Satisfiability. Apart from solving ILPs, we will also use pseudo-Boolean satisfiability (PBS) to model our problems such that they can be solved by existing software. In contrast to the former, the solution methods stem from algorithms for the Boolean satisfiability (SAT) problem and state-of-the-art solvers mostly rely on systematic backtracking that is either conflict-driven or based on a look-ahead approach [ZM02]. As SAT solvers are extremely fast and widely used, e.g., in the verification of hard- and software and VLSI, one may hope to achieve fast algorithms using this approach [VWM15]. However, SAT solvers cannot directly model optimization objectives and adding them as hard constraints (i.e., *clauses*) is usually infeasible in practice because of their size. PBS solvers, on the other hand, try to circumvent this drawback by employing techniques that are specifically tailored to constraints that consider cardinalities [RM09]. Unfortunately, compared to ILPs, it is less clear what constitutes a good model in the realm of PBS solving and we hence merely aim for small PBS models with few cardinality constraints.

1.7 Experiments and Algorithm Engineering

In this thesis, we also want to judge algorithms from a practical perspective. Practical evaluations of algorithms are important for several reasons. First, as complexity theory heavily abstracts from real-world computers, and often—also depending on the machine model—disregards implementation details, we potentially achieve more realistic predictions for running time and space demand on typical real-world instances. The same can be said w.r.t. pathological instances that may never occur in practice: The general theoretical considerations may simply be too broad to be of relevance here. In fact, already the constants that are disregarded by the usual asymptotic notation of resource demand may have a tremendous impact in practice. Second, pratical observations may well be used to reveal promising new directions for theoretical analysis, e.g., when studying the dependence on certain graph parameters or observing an unexpected approximation guarantee on a broad set of benchmark instances. Third, practical experiments do not inherently require deep theoretical considerations and are in some sense cheaper to obtain. This is of particular interest when *engineering* an algorithm, i.e., tuning the algorithm's performance in a controlled feedback-loop where one alternates between algorithm design, theoretical analysis (possibly a shallow one), thoughtful implementation, and practical evaluation. Finally, obtaining an implementation of a highly intricate algorithm serves as a proof of concept, witnessing that the algorithm's theoretical presentation is sufficiently detailed to actually understand and implement it. In fact, there are cases where theoretical errors where only discovered by attempting to implement an allegedly correct (albeit highly complex) algorithm; cf., e.g., planarity testing by Auslander and Parter [Gol63] as well as decomposition into triconnected components by Hopcroft and Tarjan [GM00].

Among many other success stories, algorithm engineering lead to staggering improvements for planarity testing, and FPT-algorithms for computing minor-closed graph measures, despite both fields essentially being exhausted from the point of classical complexity theory. See [Bar+10], particularly Chapter 6 by Chimani and Klein, for a broad introduction to algorithm engineering.

Later, we will present several successful applications of algorithm engineering where we perform preliminary benchmarks to tune our algorithms by changing their parameterizations and subroutines before finally obtaining a competitive practical algorithm. For example, in Sections 2.4 and 2.5 we are going to

- construct new ILP models for the MPS problem ($\rightarrow$ algorithm design);
- show their correctness and consider the strength of their LP relaxations ($\rightarrow$ theoretical analysis);
- choose suitable subroutines, e.g., for the separation of large constraint classes and use sophisticated existing implementations of graph frameworks and ILP solvers ($\rightarrow$ implementation); and
- perform an extensive computational study on large sets of appropriate and relevant artificial as well as real-world graphs ($\rightarrow$ practical evaluation).

1.7.1 Instance Sets

When practically evaluating algorithms for measuring the non-planarity of a graph, we always use the following two benchmark sets that are well established and widely used within the graph drawing community: On the one hand, we use the set North that contains 423 real-world non-planar graphs (5114 in total). This set was collected by Stephen C. North during his work at AT&T Bell Labs where he ran a mail service that accepted directed graphs and answered each such request with a drawing of the respective graph [Nor95]. On the other hand, we use the set Rome that contains 9249 non-planar graphs (11528 in total). The latter set was generated from a set of 112 real-world graphs stemming from the visualization of software engineering and database applications. See [Di +97] for details on the sophisticated generation procedure.

Depending on the scope of the computational evaluation and the running time of the considered algorithms, we may also use a feasible subset of the set SteinLib that was originally designed to compare solvers for problems related to Steiner trees. It contains several subsets, each corresponding to a group of either particularly hard or realistic, i.e., real-world instances [KMV00]. In addition, we will often consider a set Regular of 380 random regular graphs. Such graphs are known to be expander graphs with high probability, i.e., they are highly connected albeit being relatively sparse. We speculate that these graphs constitute particularly hard instances for measuring non-planarity. To generate this set, we used an algorithm by Steger and Wormald [SW99] that is implemented as part of the OGDF [Chi+13]. For most experiments, we use a standardized set of 20 such graphs with number n of nodes and node degree Δ for each feasible parameterization $(n, \Delta) \in \{10, 20, 30, 50, 100\} \times \{4, 6, 10, 20, 40\}$.

Finally, we will also use sets that are tailored to evaluate the specific algorithms at hand. For example, we may use much larger instances when comparing heuristics instead of exact algorithms. We will not summarize these supplemental sets here, but present them in the respective upcoming sections.

Chapter 2

Skewness and Maximum Planar Subgraphs

Cycle (*transportation*). A pedal-powered vehicle, such as a unicycle, bicycle, or tricycle, or a motorized vehicle that has either two or three wheels.

In this chapter, we consider the non-planarity measure skewness and its inverse, i.e., maximum planar subgraph. We begin by comparing various heuristics in Section 2.2 w.r.t. implementation effort, running time, and solution quality. In a second step, in Section 2.3, we turn our attention to approximation algorithms and point out obstacles that arise in several natural greedy approximation approaches. Finally, we consider exact algorithms that are based on ILP and SAT. More precisely, in Section 2.4, we study different characterizations of planarity and their merits for designing new ILP models. On the other hand, in Section 2.5, we improve the fastest such model by considering small cycles in the input. The latter constitutes a novel approach that results in surprisingly strong models and has applications for other non-planarity measures as well.

2.1 ILP Model by Mutzel

The following ILP is due to Mutzel [Mut94]. It constitutes the first exact algorithm for solving the MPS problem. Even though it was proposed more than a quarter of a century ago, it leads to surprisingly fast algorithms (cf. Section 2.4) and still is an integral part of the fastest known algorithms

for computing a graph's skewness (see Section 2.5). Jünger and Mutzel showed that both constraint classes below form facets of the planar subgraph polytope [JM96]. Given an input graph G, possibly with non-uniform edge-weight w (otherwise, we assume $w = 1$ in the following), we use solution variables $s_e \in \{0, 1\}$ for all $e \in E(G)$ that are 1 if and only if edge e is deleted, i.e., *not* in the planar subgraph. (In [Mut94], equivalent variables $x_e = 1 - s_e$ are used.) The objective minimizes the number of removed edges—thus maximizes the planar subgraph—and is given by

$$\min \textstyle\sum_{e \in E(G)} w(e)\, s_e.$$

Its optimal value directly determines $skew(G)$. For a given subset $F \subseteq E(G)$ of edges, we define $s(F) := \sum_{e \in F} s_e$ as a shorthand. Recall that by Kuratowski's theorem (cf. Section 1.4), a graph is planar if and only if it neither contains a subdivision of a K_5 nor one of a $K_{3,3}$. Hence, it suffices to ask for any member of the (exponentially large) set $\mathcal{K}(G)$ of all Kuratowski subdivisions that at least one of its edges is deleted:

$$s\big(E(K)\big) \geq 1 \qquad \forall K \in \mathcal{K}(G). \tag{2.1}$$

Clearly, constraints (2.1) are too many to be used explicitly in its entirety. Instead, in practice, we usually identify a sufficient subset of constraints via a (heuristic) separation procedure: we round the fractional solution and obtain a graph that can be tested for planarity. If it is non-planar, we extract a Kuratowski subdivision. This method does neither guarantee to always find a violated constraint if there is any, nor that the identified subdivision in fact corresponds to a *violated* Kuratowski constraint. Still, since it has these guarantees on integral solutions, it suffices to obtain an exact algorithm.

In addition to the above, we can always use Euler's bound on the number of edges in planar (bipartite) graphs:

$$s\big(E(G)\big) \geq |E(G)| - (3n - 6) + (n - 2)\mathbb{1}_{G \text{ is bipartite}}. \tag{2.2}$$

Over the years, the practical performance of this approach was improved by strong preprocessing [CG09], finding *multiple* Kuratowski subdivision in linear time [CMS07], and strong primal heuristics. The latter are presented in the following section.

2.2 Heuristics

This section is based on [CKW16]. Here, we consider heuristics to tackle the MPS problem. While solving MPS is NP-hard, there are diverse polynomial-time approaches to compute a large or maximal planar subgraph, ranging

from very trivial to sophisticated. By Euler's formula we know that already a spanning tree gives a (1/3)-approximation for MPS. Hence, all reasonable algorithms achieve this ratio. Only the *cactus algorithms* (see below) are known to exhibit better ratios. We will also consider an exact MPS algorithm based on ILPs.

All algorithms considered in this section are known (mostly for quite some time, in fact), and are theory-wise well understood both in terms of worst case solution quality and running time. To our knowledge, however, they have never been practically compared. We are in particular interested in the following quality measures, and their interplay:

– What is the practical difference in terms of running time?
– What is the practical difference in solution quality?
– What is the implementation effort of the various approaches?

Overall, understanding these different quality measures as a multi-criteria setting, we can argue for each of the considered algorithms that it is Pareto-optimal. We are in particular interested in studying a somewhat "blurred" notion of Pareto-optimality: We want to investigate, e.g., in which situations the additional sophistication of algorithms gives "significant enough" improvements. Clearly, there is a systematic weakness, as it may be highly application-dependent what one considers "significant enough". We hence cannot universally answer this question, but aim to give a guideline with which one can come to her own conclusions.

Also the measure of "implementation complexity" is surprisingly hard to concisely define, and even in the field of software-engineering there is no prevailing notion; "lines of code" are, for example, very unrelated to the intricacies of algorithm implementation. We will hence only argue in terms of direct comparisons between pairs of algorithms, based on our experience when implementing them. This measure is still intrinsically subjective (although we feel that the situation is quite clear in most cases), and opinions may vary depending on the implementor's knowledge, experience, etc. We discuss these issues when they become relevant.

As we will see in the next section, there are certain building blocks all algorithms require, e.g., a graph data structure and (except for `C` and `D`, see below) an efficient planarity test. When discussing implementation complexity, it seems safe to assume that a programmer will already start off with some kind of graph library for her basic data structure needs.[1] In the context of the ILP-based approach, we assume that the programmer

[1] Many freely available graph libraries contain linear-time planarity tests. They usually lack sophisticated algorithms for finding large planar subgraphs.

uses one of the various freely available (or commercial) frameworks. Writing a competitive branch-and-cut framework from ground up would require a staggering amount of knowledge, experience, time, and finesse. The ILP method is simply not an option if the programmer may not use a preexisting framework.

In the following section, we discuss our considered algorithms and their implementation complexity. In Section 2.2.2, we present our experimental study. We first consider the pure problem of obtaining a large planar subgraph. Thereafter, we investigate the algorithm choices when solving MPS as a subproblem in a typical graph drawing setting—the planarization heuristic.

2.2.1 Algorithms

Naïve Approach (`Nï`). The algorithmically simplest way to find a maximal planar subgraph is to start with the empty graph and to insert each edge (in random order) unless the planarity test fails. Given an $\mathcal{O}(n)$ time planarity test (we use the algorithm by Boyer and Myrvold [Joh04], which is also supposed to be among the practically fastest), this approach requires $\mathcal{O}(nm)$ overall time.[2]

In our study, we consider a trivial multi-start variant that picks the best solution after several runs of the algorithm, each with a different randomized order. The obvious benefit of this approach is the fact that it is trivial to understand and implement—once one has any planarity test as a black box.

Augmented Planarity Test (`BM`, `BM+`). Planarity tests can be modified to allow the construction of large planar subgraphs. We will briefly sketch these modifications in the context of the above mentioned $\mathcal{O}(n)$ planarity test by Boyer and Myrvold [Joh04]: In the original test, we start with a depth first search (DFS) tree and build the original graph bottom-up; we incrementally add a vertex together with its DFS edges to its children and the backedges from its decendents. The test fails if at least one backedge cannot be embedded.

We can obtain a large (though in general not maximal) planar subgraph by ignoring whether some backedges have not been embedded, and continuing with the algorithm (`BM`). If we require maximality, we can use `Nï` as a prostprocessing to grow the obtained graph further (`BM+`). While this voids

[2] We *could* speed-up the process in practice by starting with a spanning tree plus three edges. However, there are instances where the initial inclusion of a spanning tree prohibits the optimal solution and restricts one to approximation ratios $\leq 2/3$ (cf. Section 2.3).

the linear running time, it will be faster than the pure naïve approach. Given an implementation of the planarity testing algorithm, the required modifications are relatively simple per se—however, they are potentially hard to get right as the implementor needs to understand side effects within the complex planarity testing implementation.

Alternatively, Hsu [Hsu05] showed how to overcome the lack of maximality directly within the planarity testing algorithm [SH99] (which is essentially equivalent to [Joh04]), retaining linear running time. While this approach is the most promising in terms of running time, it would require the most demanding implementation of all approaches discussed here (including the next paragraph)—it means to implement a full planarity testing algorithm plus intricate additional procedures. We know of no implementation of this algorithm.

Cactus Algorithms (`C`, `C+`, `D`, `D+`). The only non-trivial approximation ratios are achieved by cactus-based algorithms [Căl+98; CS17]. Thereby, we start with the disconnected vertices of G. To obtain a ratio of 7/18 (`C`), cf. [Căl+98], we iteratively add triangles connecting formerly disconnected components. This process leaves a forest F of tree-like structures made out of triangles—*cactusses*. Finally, we make F connected by adding arbitrary edges of E between disconnected components. Since this subgraph will not be maximal in general, we can use `Nï` to grow it further (`C+`). Recently, cf. [CS17], the ratio was improved to 13/33 (`D`) by adding a new initial phase: we iteratively add diamonds (i.e., subgraphs that by addition of one edge would become a K_4), where—just as before—all vertices of each subgraph belong to pairwise different components. Naturally, we may also use this stronger approximation in combination with `Nï` to grow it further (`D+`).

From the implementation point of view, these algorithms are quite trivial and—unless one requires maximality—do not even involve any planarity testing. While a bit more complex than the naïve approach, they do not require modifications to complex and and potentially hard-to-understand planarity testing code like `BM`.

For the best approximation ratio of 4/9 one seeks not a maximal but a *maximum* cactus forest. However, this approach is of mostly theoretical interest as it requires non-trivial polynomial time matroid algorithms.

ILP Approach (`ILP`). Finally, we use the ILP by Mutzel, presented in Section 2.1, to solve MPS exactly in reasonable (but formally non-polynomial) time, see [JM96]. Its separation of Kuratowski constraints in fact constitutes

the central implementation effort. Typical planarity testing algorithms initially only answer *yes* or *no*. In the latter case, however, all known linear-time algorithms can be extended to extract a *witness* of non-planarity in the form of a Kuratowski subdivision in $\mathcal{O}(n)$ time. If the available implementation does not support this additional query, it can be simulated using $\mathcal{O}(n)$ calls to the planarity testing algorithm, by incrementally removing edges whenever the graph stays non-planar after the removal. Both methods result in a straightforward implementation (assuming some familiarity with ILP frameworks), but an additional tuning step to decide, e.g., on rounding thresholds, is necessary. The overall complexity is probably somewhere in-between `C`, `D`, and `BM`. In our study, we use the effective extraction scheme described in [CMS07] which gives several Kuratowski subdivions via a single call. We propose, however, to use this feature only if it is already available in the library: its implementation effort would otherwise be comparable to a full planarity test, and in particular for harder instances its benefit is not significant.

2.2.2 Experiments

For an exploratory study we conduct experiments on several benchmark sets. We summarize the results as follows. Observe the inversion between *F1* and *F2*.

F1. `D+` yields the best solutions. The solutions of `C+` are only marginally worse. Choosing a "well-growable" initial subgraph—in our case a good cactus—is practically important. The better solution of `BM` is a weak starting point for `BM+`; even `Nï` gives clearly better solutions.

F2. `BM` gives even better solutions than `D` and the latter already gives slightly better solutions than `C`. All three algorithms are the by far fastest approaches. Thus, if running time is more crucial than maximality, we suggest to use `BM`.

F3. `ILP` only works for small graphs. Expander graphs (they are sparse but well-connected) seem to be among the hardest instances for the approach.

F4. Larger planar subgraphs lead to fewer crossings for the planarization method. However, this is much less pronounced with modern insertion methods.

Setup & Instances. All considered algorithms are implemented in C++ (g++ 8.3.0, 64bit, `-O3`) as part of OGDF [Chi+13], the ILP is solved via CPLEX 12.6. We use an Intel Xeon Gold 6134 CPU, 3.20 GHz running Debian 10; algorithms use single physical cores, with a memory limit of 4GB per process.

Of the real-world benchmark sets presented in Section 1.7.1, we use North, Rome, and 586 graphs from SteinLib. In our plots, we group instances according to their number of nodes, rounded to the nearest multiple of 10; for Rome we only consider graphs with at least 25 nodes. Additionally, we consider two artificial sets: Regular, generated as presented in Section 1.7.1 (20 graphs for each combination of number of nodes and node degree $\Delta \in \{4, 6, 10, 20, 40\}$) but with a higher number n of nodes, chosen as $n \in \{10^2, 10^3, 10^4\}$. Secondly, we use a set BaAl of scale-free graphs that was generated following a well known model by Barabási and Albert [OMa+]. We use the same parameterization in terms of instances per number of nodes and edges as for Regular.

Our results confirm the need for heuristic approaches, as `ILP` solves less than 25% of the larger graphs of the (comparably simple) Rome set within 10 minutes. Even deploying strong preprocessing [CG09] (`+PP`) and doubling the computation time does not help significantly, cf. Fig. 2.1d. Already graphs on 30 nodes with density 3, generated like Regular, cannot be solved within 48 hours (→*F3*).

We measure solution quality by the *density* (edges per vertices) of the computed planar subgraph. See Section 2.2.2 and Tables 2.1 and 2.2 for an overview of our results. Independently of the benchmark set, `D+` achieves the best solutions, closely followed by `C+`, cf. Table 2.1 and Figs. 2.1a and 2.1b (→*F1*). We know instances where `Nï` is only a (1/3)-approximation whereas the worst ratio known for `BM+` is 2/3 (cf. Section 2.3). Surprisingly, `Nï` yields distinctly better solutions than `BM+` in practice (→*F1*).

On SteinLib, BaAl, and Regular, the algorithms `C`, `D`, and `BM` behave relatively similar w.r.t. solution quality. For Rome and North, however, `BM` yields solutions that are 20–30% better, respectively (→*F2*). This discrepancy seems to be due to the fact that the found subgraphs are generally very sparse for both algorithms on BaAl and Regular (average density of 1.1 and 1.2, respectively, for the largest graphs).

All three algorithms that do not iteratively test planarity, i.e., `C`, `D`, and `BM` are extremely (and similarly) fast; cf. Table 2.2 and Fig. 2.1c (→*F2*). For their counterparts `C+`, `D+`, and `BM+`, the `Nï`-postprocessing dominates the running time: `Nï` is worst, followed by `C+`, `D+`, and `BM+` in that order—a larger initial solution leads to fewer tries for growing. Nonetheless, we observe

that the (weaker) solutions of C and D allow for significantly more *successful* growing steps than BM ($\rightarrow$*F1*). The fact that D (D+) is barely able to achieve any solutions better than C (C+, respectively) on Rome may be explained by the low number of diamond subgraphs in this set of instances. However, even over all sets, the largest difference in solution quality between these two algorithms can be observed on SteinLib R where C achieves only 90.35% of D and C+ only 96.15% of D+ (cf. Table 2.1). In theory, the approximation ratio of C is 98.72% of that of D. However, it is still open whether the analysis of the latter is tight and we may see this as a hint towards an even better ratio of D.

Finally, we investigate the importance of the subgraph selection for the planarization method, cf. Figs. 2.1e and 2.1f. For the simplest insertion algorithms (iterative edge insertions, fixed embedding, no postprocessing, [BTT84]), a strong subgraph method (C+ or D+) is important; C and D lead to very bad solutions. For state-of-the-art insertion routines (simultaneous edge insertions, variable embedding, strong postprocessing, [CG12; CH16]) the subgraph selection is less important; even C and D become feasible.

Note that the obtained running times slightly differ from the ones in [CKW16], this is mostly due to an updated version of the OGDF and tuned implementations, most prominently noticeable for the now faster Nï. However, also the hardware that we used for our experiments was replaced. On the positive side, and as expected, our qualitative assessments from [CKW16] remain entirely valid.

It is clear from our experiments that there is a need for good heuristics or faster exact algorithms as the state-of-the-art exact algorithm is not even able to solve half of the comparably simple Rome set. In the next section, we turn our attention to better approximations. Later, cf. Sections 2.4 and 2.5, we also investigate faster exact algorithms.

2.3 Approximation of MPS and its Limits

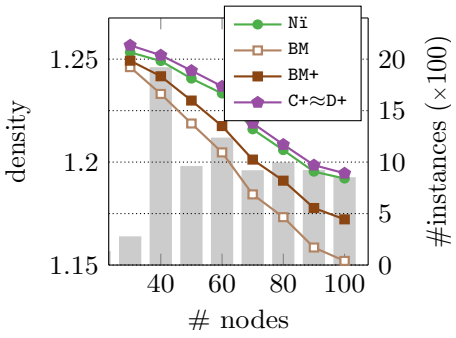
In this section, we investigate ways of achieving better approximations for MPS and highlight some key obstacles. It is based on [CHW16]. Let us begin by revisiting a well known fact: Already by Euler's formula (cf. Section 1.4), every spanning subgraph constitutes a (1/3)-approximation of MPS. Unfortunately, the same does not work for skewness and indeed no approximation for it is known. We observe that the above ratio for MPS is tight:

Table 2.1: Average density of obtained solutions in percent, relative to column-wise best algorithm. The set S (= constrained sparse) consists of PUC and SP. The set R of 1R and 2R. Number k for BaAl and Regular denotes the subset that consists of all such graphs on 10^k nodes.

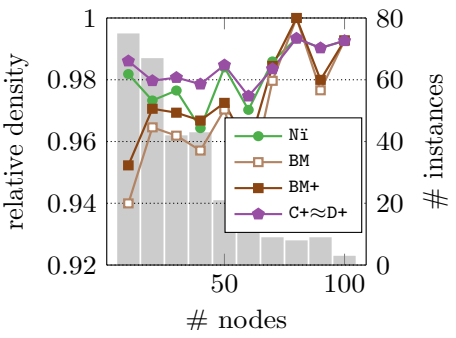
	SteinLib				BaAl			Regular		
	B-E	I*	S	R	2	3	4	2	3	4
Nï	91.24	97.40	**100**	84.62	96.42	92.27	90.57	89.66	89.39	97.25
BM	84.67	84.90	71.86	60.00	74.18	62.98	64.78	80.46	83.33	94.50
BM+	89.05	89.06	87.94	69.23	86.85	73.48	71.70	87.36	87.88	96.33
C	82.48	66.67	57.29	57.69	59.62	67.40	73.58	71.84	86.36	94.50
C+	98.54	99.48	99.50	96.15	**100**	98.90	98.74	97.70	98.48	**100**
D	85.40	70.83	59.30	63.85	64.32	71.27	77.36	76.44	88.64	94.50
D+	**100**	**100**	99.50	**100**	99.06	**100**	**100**	**100**	**100**	**100**

Table 2.2: Running time of tested algorithms in seconds. The same notation as for Table 2.1 applies.

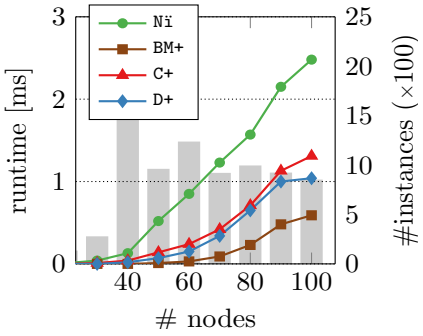
	SteinLib				BaAl			Regular		
	B-E	I*	S	R	2	3	4	2	3	4
Nï	9.84	6.31	8.57	1.04	0.02	2.48	216.73	0.02	1.56	218.74
BM	0.03	0.04	0.02	0.04	0.00	0.01	0.48	0.00	0.01	1.03
BM+	10.69	9.56	9.98	1.12	0.02	2.94	192.58	0.02	1.46	221.50
C	0.02	0.03	0.01	0.03	0.00	0.01	0.70	0.00	0.00	0.21
C+	14.34	8.84	9.60	1.07	0.02	3.25	192.42	0.02	1.67	181.39
D	0.03	0.03	0.01	0.03	0.00	0.01	2.23	0.00	0.01	0.31
D+	15.97	8.90	9.61	1.20	0.02	3.34	188.24	0.02	1.73	180.00



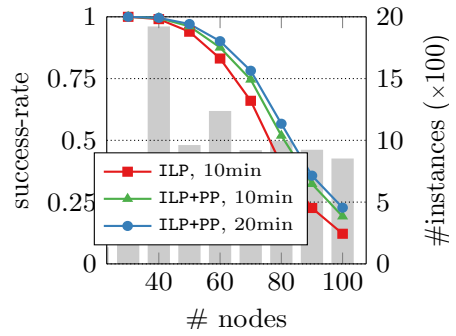
(a) ROME, solution quality, $\mathtt{C},\mathtt{D} = 1.02[1.01, 1.02]$



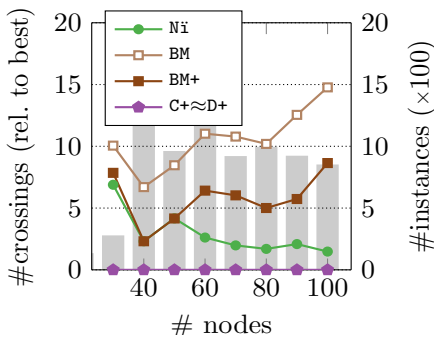
(b) NORTH, sol. quality relative to ILP (over graphs solved by ILP), $\mathtt{C},\mathtt{D} = 0.76[0.64, 0.85]$



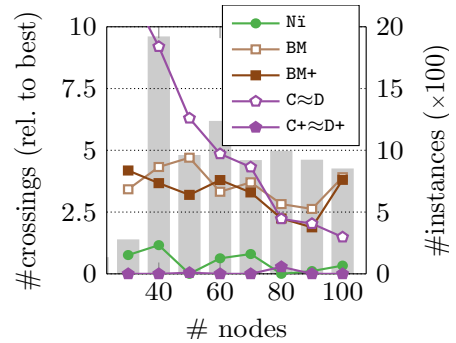
(c) ROME, running time, $\mathtt{BM},\mathtt{C},\mathtt{D} \approx 0$



(d) ROME, success-rate of ILP



(e) ROME, simple planarization, $\mathtt{C},\mathtt{D} = 102[61, 173]$



(f) ROME, state-of-the-art planarization

Figure 2.1: Running time and solution quality. Algorithms whose values are unsuitable for a plot are omitted; we list their average[min, max] over all clusters. The solution qualities of C and D, as well as C+ and D+, are too close to see any differences in the plots We hence omit D and D+ from some plots.

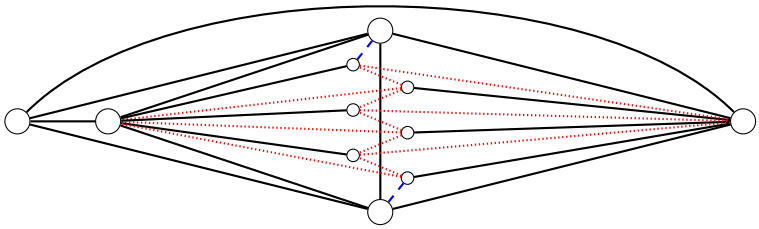


Figure 2.2: Maximal planar subgraph of G for $\ell = 6$ in Observation 2.3.1 and two edges that may be added to make it 3-connected (dashed blue). A large set of edges whose removal yields a small maximal planar subgraph is dotted red.

Observation 2.3.1. *A maximal planar subgraph of a given graph G yields an approximation ratio of at most $1/3$ for the MPS problem on G.*

Proof. Consider the complete graph K_5 on 5 nodes. We construct G by replacing a single edge vu by $\mathcal{B}^{\ell}_{v,u}$, and adding a Hamiltonian path $P = p_1, ..., p_\ell$ for the nodes $\mathcal{I}(\mathcal{B}^{\ell}_{v,u})$. Let $S \coloneqq E(K_5) \setminus \{vu\} \cup \{vp_k \mid k \text{ odd}\} \cup \{p_k u \mid k \text{ even}\}$. S is a maximal planar subgraph of G since adding any edge yields a K_5 subdivision (cf. Fig. 2.2). An MPS H can be obtained by removing any one edge outside of $\mathcal{B}^{\ell}_{v,u}$ from G. The approximation ratio is thus at most $\lim_{\ell \to \infty} |S|/|E(H)| = 1/3$. □

A standard preprocessing technique, the non-planar core (NPC), may be used to reduce the above graph G to a weighted K_5, thereby preventing a ratio of less than $2/3$. However, we may amend G to prevent any such preprocessing. This is, for example, achieved simply by making the instance 3-connected by addition of two new edges, each connecting one of the top or bottom center nodes with its closest internal one in the drawing (see dashed edges in Fig. 2.2).

2.3.1 Algorithms Inspired by Planarity Tests

First, we focus on DFS- and breadth first search (BFS)-based algorithms, providing hardness results and bounds for families of approximation algorithms. We denote the problem of finding a maximum planar subgraph that contains a given DFS (or BFS)-tree by *MPS-DFS* (or *MPS-BFS*, respectively). In particular, any known algorithm based on a planarity test in fact solves MPS-DFS heuristically.

A *k-book* is a collection of k half-planes (*pages*) that share a common boundary (*spine*). A *k-book embedding* is an embedding of a graph into a k-book such that the vertices are placed on the spine, every edge is drawn on a single page, and no two edges cross each other. Consider a circle with straight-line chords $\mathcal{C}$. A *circle graph* is the intersection graph of the latter: $\mathcal{C}$ are its nodes, two nodes are adjacent if and only if their chords cross. The *overlap graph* is the graph where each chord is an edge, and the chords' end nodes are connected by a Hamiltonian cycle according to the original drawing. For a given circle graph $G = (V, E)$ and $c, k \in \mathbb{N}$, the problem of finding a subset $V' \subseteq V$, such that $|V'| \geq k$ and $G[V']$ is c-colorable is the c-Colorable Induced Subgraph problem for Circle Graphs (c-CIG). It is NP-hard for $c \geq 2$ [CL91]. Clearly, any solution for c-CIG corresponds to a c-book embedding of the respective overlap graph; the circle corresponds to the spine and each color class is embedded in its own page.

Theorem 2.3.2. *MPS-DFS is* NP*-hard. In particular, for each function f that assigns each graph a start node for its DFS, there exists an (infinite) family $\mathcal{G}$ of tuples, each consisting of a graph G and a DFS-tree on G that satisfies f, such that MPS-DFS remains* NP*-hard on $\mathcal{G}$.*

Proof. We perform a reduction from 2-CIG to MPS-DFS. Let (G, k) be an instance for 2-CIG and $C = (W, F)$ be the corresponding overlap graph. Let $n \coloneqq |W|$, $m \coloneqq |F|$, and $\pi \colon [n] \to W$ denote the cyclic order of W induced by C. Let $B_i \coloneqq \mathcal{B}^m_{\pi_i,\pi_{i+1}}$ denote a π_i-π_{i+1}-bundle of thickness m and $B'_i \coloneqq B_i \cup \{\pi_i\pi_{i+1}\}$. We construct the input graph $D \coloneqq (\bigcup_{i\in[n]} V(B_i), F \cup E_\mathrm{B})$ for MPS-DFS, with $E_\mathrm{B} \coloneqq \bigcup_{i\in[n]} E(B'_i)$; see Fig. 2.3. The set $T \coloneqq \{\pi_i\pi_{i+1}, u\pi_{i+1} \mid 0 \leq i < n-1, u \in \mathcal{I}_i\} \cup \{u\pi_{n-1} \mid u \in \mathcal{I}_{n-1}\}$, where $\mathcal{I}_i \coloneqq \mathcal{I}(B_i)$, is a DFS-tree of D.[3] We show that the 2-CIG instance (G, k) has a solution if and only if D has a planar subgraph of size $\xi \coloneqq k + n(2m + 1)$ that contains T.

(If) Assume there is a planar embedded subgraph S of D that contains T and has ξ edges. D contains $m + n(2m + 1)$ edges. Removing more than m edges from D yields a graph with less than ξ edges. Thus, there are at least $m + 1$ edges from each B'_i in S. Consequently, for each pair of nodes π_i, π_{i+1} there is a path within B'_i connecting them. Hence we have a cycle through $\pi_0, \pi_1, ..., \pi_{n-1}, \pi_0$, splitting $E(S) \setminus E_\mathrm{B}$, the edge set of S corresponding to chords, into an inside and an outside partition. Since S is planar, this induces a 2-book embedding of those edges. Thus, we have a solution of 2-CIG on (G, k) as $|E(S)| - |E_\mathrm{B}| = k$.

[3] We start at π_0 with $\pi_0\pi_1$. Next, we pick all edges of B_0 that are incident to π_1 since the $\mathcal{I}(B_0)$-vertices lead only to π_0 (visited). We iterate this until we arrive at $\pi_{n-2}\pi_{n-1}$. Finally, we pick all edges connecting π_{n-1} with $\mathcal{I}(B_{n-2})$ and $\mathcal{I}(B_{n-1})$.

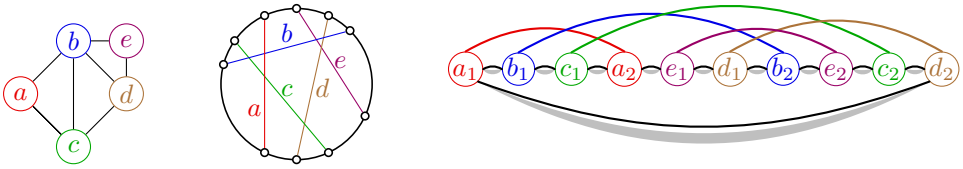


Figure 2.3: Constructions for the proof of Theorem 2.3.2. The circle graph G on the left with the respective overlap graph in the middle and a schematic depiction of the input graph D for MPS-DFS with ordering $\pi_0 = a_1, \pi_1 = b_1, ...$ on the right (bundles sketched in gray).

(Only If) Assume there is a solution for 2-CIG on (G, k). This corresponds to a 2-book embedding of a subgraph $C' \coloneqq (W, E')$ of C, where the vertices W are placed on the spine according to π, and $|E'| \geq k$. Adding E_B to C' yields a planar graph. Note that $T \subseteq E_\mathrm{B}$ and C' contains $|E_\mathrm{B}| + k \geq \xi$ edges.

Note that the proof works independently of the DFS start node since π is cyclic and π_0 can be chosen arbitrarily. □

We will see that any algorithm adding edges to an arbitrary DFS-tree has an approximation ratio of at most 2/3. However, the second part of the theorem above shows that we cannot simply iterate over all possible start nodes to find a tractable MPS-DFS instance and use this to approximate MPS.

Theorem 2.3.3. *An optimal solution to MPS-DFS yields an approximation ratio of at most* 2/3 *for the corresponding MPS problem.*

Proof. Given a number $p \geq 4$, consider the following graph $G \coloneqq (V, S \cup \{\tilde{e}\} \cup T)$ with $V \coloneqq \{u_1, ..., u_p, v_1, ..., v_p\}$, and $S \coloneqq \bigcup_{i=1}^{p-1} \{u_i u_{i+1}, v_i v_{i+1}\}$. The edges in S form two disjoint paths, both of length $p - 1$. Let T be an edge set that triangulates $G[S]$. Note that this is possible (cf. Fig. 2.4) in a way such that

$$\forall e \in T \colon e = u_i v_j \wedge |i - j| \leq 2. \tag{2.3}$$

Finally, we define $\tilde{e} \coloneqq u_p v_1$. Observe that $|T| = 4p - 4$ and $P \coloneqq S \cup \{\tilde{e}\}$ forms a Hamiltonian path. Assume that the DFS on G returns P. We prove that any planar subgraph H of G that contains P can have at most half of the edges of T.

Any such graph can be constructed by successively inserting edges of T into $G[P]$. After each step the there are at least two faces f_1, f_2 that have exactly one edge of T on their boundary: Initially, adding the first edge to

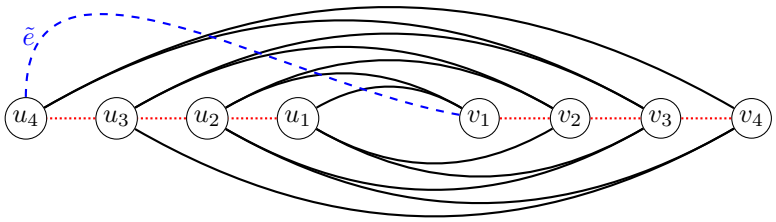


Figure 2.4: Drawing of the graph G with $p = 4$ in the proof of Theorem 2.3.3. Edges of S are dotted red. The edge $\tilde{e}$ is dashed blue.

$G[P]$ yields two such faces. If the next edge is embedded neither in f_1 nor in f_2 the invariant holds. Otherwise, the edge is embedded in, say, f_1. Then f_1 is split into two faces, one of which becomes the new f_1. For each edge $t \in T$, $P \cup \{t\}$ has a cycle of length at least $p-2$, which follows from eq. (2.3) by construction of P. We conclude that H has two faces of degree at least $p-2$, and at least $2p-10$ edges are missing for H to be a triangulation.

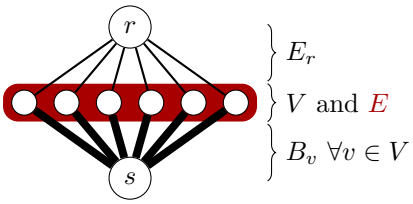
The edges $E(G) \setminus \{\tilde{e}\}$ form an MPS. It follows that MPS-DFS approximates MPS by a ratio of at most $\lim_{p\to\infty}(|P|+|T|-(2p-10))/|E(G)\setminus\{\tilde{e}\}| = 2/3$. □

The result above shows that the approximation ratio of DFS-based algorithms is bounded from above by 2/3. We wonder if this is caused by the special structure of DFS-trees or if this can be extended, for example, to BFS-based algorithms:

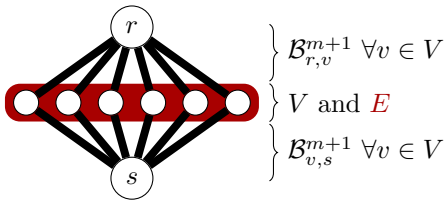
Theorem 2.3.4. *MPS-BFS is* NP*-hard. In particular, for each function f that assigns each graph a start node for its BFS, there exists an (infinite) family $\mathcal{G}$ of tuples, each consisting of a graph G and a BFS-tree on G that satisfies f, such that MPS-BFS remains* NP*-hard on $\mathcal{G}$.*

Proof. We give a reduction from HC to MPS-BFS. Let $G = (V, E)$ be an instance for HC, $n := |V|$, $m := |E|$, let s denote a new node, and $B_v := \mathcal{B}^{m+1}_{v,s}$ for each $v \in V$. We construct an input graph $G' := (V', E')$ for MPS-BFS, where $V' := \{r\} \cup V \cup V_B$, $V_B := \bigcup_{v\in V} V(B_v)$, $E' := E_r \cup E \cup E_B$, $E_r := \{rv \mid v \in V\}$, and $E_B := \bigcup_{v\in V} E(B_v)$, cf. Fig. 2.5(a). G' contains $2 + n(m+2)$ nodes and $m + n(2m+3)$ edges. Choose $u \in V$ and $\tilde{p} \in \mathcal{I}(B_u)$ arbitrarily. We define $T := \{\tilde{p}s\} \cup E(G'[V' \setminus \{s\}]) \setminus E$, a BFS-tree of G'.[4]

[4]Starting at r (level 0) includes all edges of E_r. E cannot be taken since all of V lies on level 1. Each node $v \in V$ is connected to all of $\mathcal{I}(B_v)$, which lie on level 2. Only s remains, which is connected to $\tilde{p}$—the first investigated node on level 2.



(a) Schematic drawing of G' in the proof of Theorem 2.3.4.



(b) Schematic drawing of G' in the proof of Theorem 2.3.7.

Figure 2.5: Similar constructions for the proofs of Theorems 2.3.4 and 2.3.7, both examples for $|V(G)| = 6$. Bold edges depict bundles of $m + 1$ parallel 2-paths. Any optimal solution removes only edges from E (red).

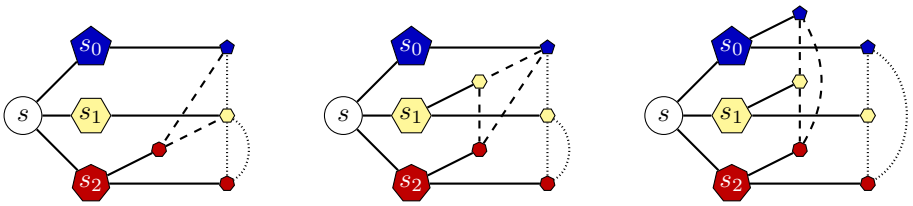
We show that G has a HC if and only if G' has a planar subgraph of size $\xi := n(2m + 4)$ that contains T.

(If) Given a planar subgraph H of G' with ξ edges that contains T, there are at most $m - 1$ edges of G' not in H since $|E'| - m < \xi$. Thus, for each bundle at least one 2-path is part of H. It follows that there can be at most n edges of E in H since H is planar. Consequently, $|E(H)| \leq k-m+|E'|$ where $k := |E \cap E(H)| \leq n$. Assuming $k < n$ leads to $|E(H)| < \xi$, a contradiction. By planarity of H we observe that $H[V]$ forms a Hamiltonian cycle in G.

(Only if) Given a Hamiltonian cycle C on G. We construct a planar subgraph $H := G'[T \cup C \cup E_B]$ that contains T (by construction) and has ξ edges. Note that adding C to T yields a planar graph since $H[\{r\} \cup V]$ forms a wheel graph. Likewise, adding E_B preserves planarity since $G'[E_B]$ is planar and contains a face with all nodes of V that allows an arbitrary ordering of those nodes.

Finally, we show the independence of the start node. From the above input graph G', we construct G'' by replacing each edge $rv \in E_r$ with $\mathcal{B}^{m+1}_{r,v}$ (cf. Fig. 2.5(b)), and replacing each edge of E with a path containing five new edges where each of the four new nodes is also connected to r by a new edge. Note that we can reach all nodes of V in at most four BFS-levels, independent of the start node. Consequently, none of the 5-paths that correspond to edges in E can be fully contained in the resulting BFS-tree. We conclude that any BFS-tree constructed in the above way allows a reduction from HC to MPS-BFS. □

As for DFS-trees, we also have that any algorithm adding edges to an arbitrary BFS-tree has an approximation ratio of at most 2/3:



(a) Two common nodes. (b) One common node. (c) Disjoint triangles.

Figure 2.6: Arising $K_{3,3}$-subdivisions after adding two triangles to the BFS-tree in the proof of Theorem 2.3.5. One triangle is dotted, the other is dashed.

Theorem 2.3.5. *An optimal solution to MPS-BFS yields an approximation ratio of at most* $2/3$ *for the corresponding MPS problem.*

Proof. Let $G = (V, E)$ denote a triangulated graph that allows a 3-coloring $\phi\colon V \to [3]$ of the nodes, for example an even cycle C with two new nodes adjacent to all of C. We define the input graph $G' := (\{s, s_0, s_1, s_2\} \cup V, E \cup T)$ for MPS-BFS with $T := \{ss_i \mid i \in [3]\} \cup \{s_{\phi(v)}v \mid v \in V\}$. T is a BFS-tree rooted at s. By construction, every triangle in G' requires 3 nodes of V of different color. We can add at most one triangle to T, as a $K_{3,3}$-subdivision would arise otherwise, see Fig. 2.6. Hence, the number of triangular faces in any planar subgraph H of G' that contains T is bounded by a constant, independent of $|V|$. Thus, the upper bound on the approximation ratio converges from above to $2/3$ for large $|V|$. □

Since any DFS or BFS-tree is also a spanning tree, we have:

Corollary 2.3.6. *It is* NP*-hard to find a maximum planar subgraph that contains a given spanning tree. Likewise, an optimal solution to this problem approximates MPS with at most* $2/3$.

2.3.2 MPS is NP-hard: A Simple Proof

Inspired by our proof that MPS-BFS is NP-hard, we can give a shorter proof for the hardness of MPS itself. Liu and Geldmacher [LG79] gave a 2-step-reduction of Vertex Cover (VC) to a HC restricted to triangle-free graphs and from that to MPS. We give a direct simple reduction from general HC to MPS.

Theorem 2.3.7. *MPS is* NP*-hard.*

Algorithm 1: Cactus Algorithm

Input: connected simple graph $G = (V, E)$

```
1 edge set S_1 := ∅
2 while ∃ triangle T ⊆ E whose nodes are in 3 different components of
    (V, S_1) do
3   S_1 := S_1 ∪ T
4 S_2 := S_1
5 while ∃ edge e ∈ E whose nodes are disconnected in (V, S_2) do
6   S_2 := S_2 ∪ {e}
7 return S_2
```

Proof. Let $G = (V, E)$ be an instance for HC, $n := |V|$, and $m := |E|$. We construct an input graph G' for MPS by adding two nodes r, s and the edge set $E_{\mathrm{B}} := \bigcup_{v \in V} (\mathcal{B}^{m+1}_{r,v} \cup \mathcal{B}^{m+1}_{v,s})$ (cf. Fig. 2.5(b)). Note that G' contains $2 + n(2m + 3)$ nodes and $m + 4n(m + 1)$ edges. We show that G has a Hamiltonian cycle if and only if G' has a planar subgraph of size $\xi := |E_{\mathrm{B}}| + n$.

(If) Given a planar subgraph H of G' with ξ edges. There are at most $m - 1$ edges of G' not in H since $|E(G')| - m < \xi$. Thus, for each bundle in E_{B} at least one 2-path is part of H. It follows that there can be at most n edges of E in H as H is planar. Consequently, $|E(H)| \leq k - m + |E(G')|$ where $k \leq n$ equals the number of edges of E in H. Assuming $k < n$ leads to $|E(H)| < \xi$, a contradiction. By planarity of H we observe that $H[V]$ forms a Hamiltonian cycle in G.

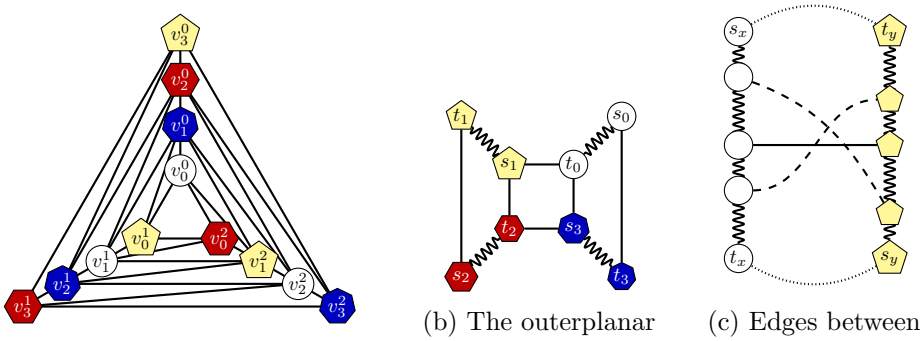
(Only if) Given a Hamiltonian cycle C on G. The graph $H := G'[C \cup E_{\mathrm{B}}]$ has ξ edges and is planar. □

Consider an MPS instance $G = (V, E)$. We can construct G' by replacing every edge in E with a path of length $k := \lceil p/3 \rceil$. Now all cycles in G' contain at least p nodes, and $\mathrm{OPT}(G') = \mathrm{OPT}(G) + (k - 1)|E|$. We conclude:

Corollary 2.3.8. *MPS remains* NP*-hard for graphs with any given girth.*

2.3.3 Algorithms Inspired by Cactus Structures

The (greedy) *Cactus Algorithm*, see Alg. 1, for MPS was developed by Călinescu et al. [Căl+98] and first constructs a cactus subgraph S_1 consisting of triangles joined at single nodes. The resulting structure S_2 achieves a tight approximation ratio of 7/18. When the first phase of the algorithm is replaced to find a cactus structure of maximum cardinality (which requires



(a) The graph H_p for $p = 4$.

(b) The outerplanar graph X_p on $V(H_p)$.

(c) Edges between V_x and V_y in X_p.

Figure 2.7: Constructions in the proof of Theorem 2.3.9.

the use of a graphic matroid parity subalgorithm), the approximation ratio can be improved to 4/9. One may either search for an algorithm with a better approximation guarantee or for an algorithm with an approximation ratio better than 7/18 that requires only simple operations (in contrast to the matroid-based algorithm), possibly again based on a greedy scheme. Poranen proposed two algorithms that greedily select triangles and conjectured approximation ratios of at least 4/9 [Por08]. However, both conjectures were refuted by Fernandes et al. [FC07, Sect. 56.6]. We show that related, more general classes of algorithms are not suited to achieve the desired approximation guarantee or have an approximation ratio of at most 1/2.

It is fairly natural to ask for a more sophisticated yet easily implementable greedy selection of the triangles to build a cactus. We first investigate algorithms that greedily select either edges or triangles in an "intuitively smart" manner. Given a graph G and a subgraph $G' \subseteq G$, we say that an edge $e \in E(G)$ is *forbidden* in G' if and only if $G' + e$ is non-planar. Similarly, we call an edge set $F \subseteq E(G)$ forbidden if and only if there is a forbidden edge $f \in F$.

The algorithm that iteratively picks an edge (or triangle) that minimizes the number of resulting forbidden edges (or triangles), is called *Greedy Edge Selection* (GES) (or *Greedy Triangle Selection* (GTS), respectively).

Theorem 2.3.9. *GES has a tight approximation ratio of* 1/3.

Proof. Let $p \geq 4$. Define $H_p := (V, E_H)$ with $V := \{v_\ell^i \mid \ell \in [p], i \in [3]\}$ and $E_H := \{v_\ell^i v_\ell^{i+1} \mid 0 \leq \ell \leq p-1, i \in [3]\} \cup \{v_{\ell-1}^i v_\ell^i \mid 1 \leq \ell \leq p-1, i \in [3]\} \cup \{v_\ell^i v_{\ell-1}^{i+1} \mid 1 \leq \ell \leq p-1, i \in [3]\}$, cf. Fig. 2.7(a). We define $\Lambda(v_\ell^i) := \ell$ as the

level of v_ℓ^i. Note that H_p is a triangulation and 4-colorable with the coloring $\phi(v_\ell^i) \coloneqq (3\ell + i) \bmod 4$. For any color $c \in [4]$, let $V_c \coloneqq \{v \in V \mid \phi(v) = c\}$ be the c-colored node partition induced by ϕ. We denote the increasing order of V_c according to Λ by π^c. For each of the four colors we define the (new) path $P_c \coloneqq \{\pi_i^c \pi_{i+1}^c \mid 1 \leq i < |V_c|\}$. The lowest and highest level node of a path P_c is denoted by s_c and t_c, respectively. We obtain the graph X_p on the nodes V by adding $\{t_0s_1, s_1t_2, t_2s_3, s_3t_0, s_0t_3, t_1s_2\}$ to the paths P_c, cf. Fig. 2.7(b).

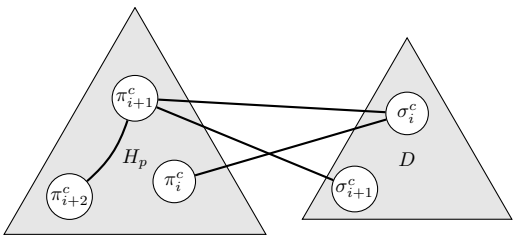
Consider the graph $G \coloneqq H_p \cup X_p$ (over the common node set V) as our input. The triangulation H_p is an MPS of G. The graph X_p is outerplanar. Thus, we can add any single edge planarly to X_p, and X_p can arise during GES since none of its edges was forbidding any other edges. By showing that we can only add a constant number of edges to X_p while preserving planarity we bound the approximation ratio by $\lim_{p\to\infty}(|E(X_p)|+\text{const})/(|E(H_p)|) = 1/3$.

We can ignore all edges incident to nodes $\{s_c, t_c \mid c \in [4]\}$: this is a constant number of edges since we have bounded degree (independent of p). Given two colors x, y, there are at most two faces in any embedding of X_p that have P_x and P_y on their boundary. Traversing any such face will visit the nodes along both paths in the same order (either $s_x \to t_x$ and $s_y \to t_y$; or $t_x \to s_x$ and $t_y \to s_y$). Let $E_{xy} \subseteq (V_x \times V_y) \cap E_H$ be an arbitrary set of independent edges whose endpoints are non-adjacent in X_p. The orderings π^x and π^y induce two orderings of E_{xy}. By construction of H_p we have $|\Lambda(v) - \Lambda(w)| \leq 1$ for all $vw \in E_H$. Hence, we observe that the above two orderings of E_{xy} are in fact identical. It follows that we can insert at most one edge of E_{xy} into each of the at most two suitable faces of X_p, cf. Fig. 2.7(c). The number of color pairs is constant. Thus, for any color pair (x, y) and suitable face f, the insertable edges $E'_{xy} \subseteq (V_x \times V_y) \cap E_H$ need to be either adjacent, or incident to adjacent nodes. Since G has bounded degree, we can only add a constant number of edges to X_p. □

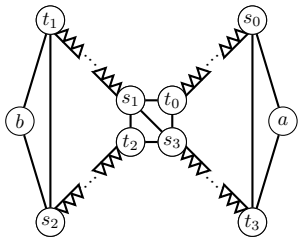
Theorem 2.3.10. *Any algorithm that selects the edges picked by GTS has an approximation ratio of at most* $7/18$.

Proof. Let G be the graph of the proof of Theorem 2.3.9 for an arbitrary but fixed $p \geq 5$, and $n_p \coloneqq |V(G)| = 3p$. Again, we speak of the paths P_c for the colors $c \in [4]$, and the outerplanar subgraph X_p of G. Our initial argument is based on the same principle as before. Coarsely speaking we replace the edges of the paths P_c by new triangles, preserve outerplanarity, and extend H_p by a similar structure on the newly inserted nodes:

Let D be a copy of H_{p-1} where we delete the node v_{p-2}^2. Note that D is triangulated with the exception of one face of degree 5. As in the



(a) Schematic structure of G' showing only some nodes of color c.



(b) The outerplanar graph X'_p.

Figure 2.8: Construction in the proof of Theorem 2.3.10.

proof above, this graph is 4-colorable which induces the node partitions $D_c \coloneqq V_c(D)$ for $c \in [4]$. D can be seen as a copy of H_p where one node of each color $(v^0_{p-1}, v^1_{p-1}, v^2_{p-1}, v^2_{p-2})$ is removed. We keep the notation of the ordering of nodes V_c in X_p by π^c and denote the analogous ordering of the nodes in the newly introduced partitions D_c by σ^c. Let $X'_p \coloneqq (V(X_p) \cup V(D) \cup \{a, b\}, E(X_p) \cup E_\triangle \cup \{s_1s_3, s_0a, at_3, t_1b, bs_2\})$ with $E_\triangle \coloneqq \{\pi^c_i\sigma^c_i, \sigma^c_i\pi^c_{i+1} \mid c \in [4], \pi^c_i\pi^c_{i+1} \in P_c\}$, see Fig. 2.8. I.e. $E_\triangle$ consists of a level-monotone Hamiltonian path for each color class.

Let $G' \coloneqq (V(X'_p), E(X'_p) \cup E(H_p) \cup E(D))$. The graph $J \coloneqq H_p \cup D$ is a planar subgraph of G'. Every edge in X'_p is part of a triangle and the graph remains outerplanar. Thus, we can add any single triangle planarly to X'_p, and X'_p could arise during GTS on G'. Analogous to the proof for Theorem 2.3.9, we can only add a constant number of edges to X'_p while preserving planarity.

Let F_J denote the set of triangular faces in J. We obtain the graph G'' from G' by inserting new nodes v_f of degree 3 for all $f \in F_J$, connecting v_f with the nodes on the boundary of f. Let $L \coloneqq \{v_f \mid f \in F_J\}$ denote the newly inserted nodes and E_L the incident edges. Considering G'' as the input for GTS, similar to above, the number of edges that we can add to X'_p while preserving planarity is bounded by $|L| + \text{const}$: Any edge in E_L is part of a 2-path u_1-w-u_2, where $u_i \in V(G')$, $\phi(u_1) \neq \phi(u_2)$, and $w \in L$. On the other hand, $J \cup (L, E_L)$ remains planar. We conclude that the approximation ratio is at most

$$\lim_{p\to\infty} \frac{|E(X'_p)| + |L| + \text{const}}{|E(J)| + |E_L|} = \lim_{p\to\infty} \frac{(3n_p + \text{const}) + (4n_p + \text{const}) + \text{const}}{(6n_p + \text{const}) + (12n_p + \text{const})}.$$

□

Algorithm 2: Dense Subgraph Selection (DSS)

Input: parameter $k \in \mathbb{N}_{\geq 3}$, connected simple graph $G = (V, E)$

edge set $S \coloneqq \varnothing$
while S is not maximal planar **do**
  Find a planar subgraph Q with up to k nodes W such that
    (i) $S[W] \subsetneqq E(Q)$,
    (ii) $S \cup E(Q)$ is planar,
    (iii) Q has max. density among all subgraphs that satisfy (i–ii), and
    (iv) possibly further restrictions (see text).
  $S \coloneqq S \cup E(Q)$
return S

Corollary 2.3.11. *Any algorithm that first selects an arbitrary (possibly maximal) set of triangles has an approximation ratio of at most* 7/18*.*

Observe that this bound matches the one of Alg. 1 [Căl+98]. Similar to any DFS- and BFS-based algorithms, it remains NP-hard to determine a maximum set of edges that can be added planarly to a selected set of triangles. We will show a more general result in Theorem 2.3.12.

We investigate the selection of dense subgraphs, which is a natural generalization of triangle-based algorithms such as GTS. Given an edge set S and a node set W, we define $S[W]$ as the edges of S that connect nodes of W. Let the *density* of a graph (V, E) be defined as $|E|/|V|$, the edges per node.

We denote Algorithm 2 by DSS. In its most general form (DSS-U) we do not pose any further restrictions (iv) on the selection of dense subgraphs: they may overlap arbitrarily. A restricted version of this algorithm, called DSS-D, requires the subgraphs Q in the loop to be node disjoint to the current structure S. Similarly, we denote by DSS-C the algorithm with the restriction that the nodes of Q are pairwise disconnected in the current structure S.

Theorem 2.3.12. *Consider any MPS instance G. It remains* NP*-hard to find a maximum planar subgraph of G under either the restriction that it contains* (a) *the solution S of DSS-D, or the restriction that it contains* (b) *the solution S of DSS-C, respectively.*

Proof. (a) Given an arbitrary triangle-free graph $G = (V, E)$, we construct G' by adding $k - 1$ nodes V_v for each $v \in V$, such that $G_v \coloneqq G'[\{v\} \cup V_v]$

is triangulated. Let $S \coloneqq \bigcup_{v \in V} E(G_v)$. Note that each G_v is a graph on k nodes with maximal density and that any other subgraph of G' has strictly lower density. Consequently, the algorithm selects each G_v to S. Thus, any subgraph H' of G' that contains S corresponds to a subgraph H of G with $|E(H')| = |E(H)| + n(3k-6)$. MPS is NP-hard on triangle-free graphs, see Corollary 2.3.8.

(b) Consider a graph G together with a spanning tree T. We know from Corollary 2.3.6 that it is NP-hard to find a maximum set of edges that can be added planarly to T. Replacing each edge of T with a triangulated subgraph on k nodes in G yields an instance where Alg. 2 can select exactly the structures corresponding to T. Thus, finding a maximum set of edges that can be added to the selected structure remains NP-hard, independent of k. □

Note that Theorem 2.3.12 for DSS-C and $k = 3$ is the above claimed hardness result for algorithms based on triangle selection.

Theorem 2.3.13. *For any fixed $k \geq 3$, DSS-U has an approximation ratio of at most $1/2$. For any fixed $k \geq 7$ any variant of DSS that poses arbitrary restrictions (iv) on the cut of Q with S has an approximation ratio of at most $1/2$.*

Proof. First assume that $k \geq 7$. Let $F \coloneqq \{f_0, ..., f_3\}$ denote the set of faces of a K_4, δ_i the set of nodes incident to face f_i and $\kappa \coloneqq k - 7$. We define $F' \coloneqq F \setminus \{f_0\}$ and $\{b, t, u_0\} =: \delta_0$. We construct $G = (V, E)$ with $V \coloneqq V(K_4) \cup \{w_i \mid f_i \in F'\} \cup \{u_{i+1} \mid i \in [\kappa]\}$, $E \coloneqq E(K_4) \cup E_W \cup E_U$, $E_W \coloneqq \{w_i v \mid f_i \in F', v \in \delta_i\}$, and $E_U \coloneqq \bigcup_{i=1}^{\kappa} \{bu_i, u_i t, u_i u_{i-1}\}$. Note that G is triangulated, planar, and contains exactly k nodes. Furthermore, we cannot connect any nodes w_i, w_j, $i \neq j$, while preserving planarity. We define the input graph G' as $(V \cup L, E \cup E_L)$, where $L \coloneqq \{s_1, ..., s_\ell\}$ and $E_L \coloneqq \bigcup_{i \in [\ell]} \{s_i w_1, s_i w_2, s_i w_3\}$ (cf. Fig. 2.9), for some $\ell \geq 7$.

The algorithm may pick a graph Q that is the entire triangulated subgraph G in its first iteration, since G contains exactly k nodes. Thus, nodes in L can only be added with a single edge and we thus pick at most $1/3$ of E_L. On the other hand, a planar subgraph $H \subseteq G$ can be obtained by picking every edge in E except for the edge of K_4 incident to f_1 and f_2. Then, each node in L can be connected with w_1 and w_2 (picking $2/3$ of E_L), giving $H' \subseteq G'$. We conclude that the approximation ratio is at most $\lim_{\ell \to \infty}(|Q| + \ell)/|H'| = \lim_{\ell \to \infty}(|E| + \ell)/(|E| - 1 + 2\ell) = 1/2$.

For $k < 7$ we construct the graph G as for $k = 7$ where DSS-U may still return a subgraph containing G, independent of k and ℓ. □

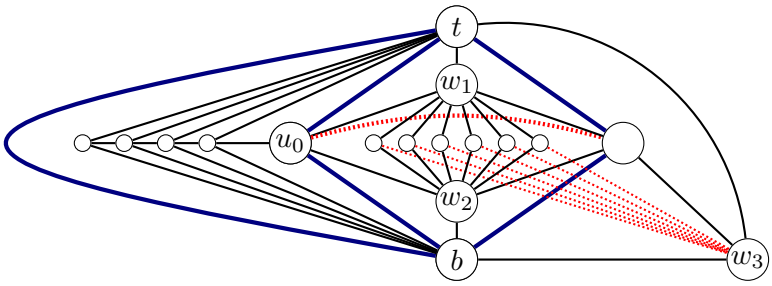


Figure 2.9: Schematic drawing of the input graph for $\ell = 6$ and $k = 4$ in the proof of Theorem 2.3.13. The K_4 subgraph is highlighted by bold edges (blue). Dotted edges (red) are not included in H_2.

We conclude that better approximations of MPS seem difficult to obtain and intuitively promising directions, such as DFS-based algorithms, inherently present us with new but similar challenges.

We hence want to investigate *exact* algorithms for MPS more closely and start by considering models based on other planarity criteria in the next section.

2.4 ILP Models by Planarity Criteria

In this section, we focus on exact algorithms for solving MPS. It is based on [CHW19]. Our aim is to develop new models that can be solved quickly by state-of-the-art PBS or ILP solvers. As such, each of the algorithms that we consider in this section is either based on PBS- or ILP-solving. To this end, we consider other established criteria for planarity that do not rely on Kuratowski subdivisions.

All models are presented as ILPs. Since their PBS counterparts directly map to the ILPs where clauses naturally correspond to constraints, we do not explicitly list the PBS formulations. We highlight optional constraints that we include in the hope to help quickly finding strong dual bounds with the symbol $\star$.

For our first model, that is the basis of our comparison, we use the traditional ILP by Mutzel, with optional ($\star$) Euler constraints (2.2), that we presented in Section 2.1. Recall that this means we use solution variables $s_e \in \{0, 1\}$ for each $e \in E(G)$ that are 1 if and only if edge e is removed, i.e., not in the planar subgraph. We tune its implementation as follows.

Algorithm engineering & preliminary benchmarks of known model. Using an ILP solver, we separate on LP-solutions by rounding the computed fractional values, thus obtaining a graph $H \subset G$ and extracting Kuratowski subdivisions from H. Our experiments indicate that rounding down values that are smaller than 0.99 (and 0.9 in a second round), yields locally optimal (w.r.t. the algorithm's parameter space) results. For each LP-solution $\tilde{s}$, we extract up to 250 Kuratowski subdivisions in linear time. We use a heap to collect 50 most violated (w.r.t. $\tilde{s}$) constraints from the extracted ones and add them to the ILP. For the PBS solver, we iteratively search for satisfying variable assignments and check each for planarity, adding up to 50 lazy Kuratowski constraints each.

Common basis. For each new model below, we start with the above program by Mutzel but omit the Kuratowski constraints (2.1).

2.4.1 Facial Walks

Recall that for any connected planar graph, there is an embedding Π, represented by a cyclic order of edges around the nodes that corresponds to a planar drawing of the graph. The regions bounded by the edges are the faces of Π (cf. Definitions 1.4.3, 1.4.5 and 1.4.6). The facial walk model is based on an idea developed in [Bey+16] to compute the genus of a graph; it constitutes the only known model for the latter problem and simulates the face tracing algorithm that proceeds as follows on a given embedding Π: Starting at a node v, we traverse an edge vw and continue with the succeeding edge at w (w.r.t. the rotation at w in Π). This is iterated until we again arrive at vw, closing the traced face's boundary. By repeating this operation until all edges are traversed exactly once in both directions, we establish all faces of Π. We simulate this algorithm by introducing variables that map each arc to its face. This allows us to employ Euler's polyhedron formula (1.4.7), relating the number of faces in any planar drawing of a graph in the plane with the number of nodes as well as the number of edges. Overall, this establishes the planarity of the subgraph.

Let $\bar{f}$ be an upper bound on the number of attainable faces. In our experiments, we set $\bar{f} \coloneqq \min\{2n-4, m-n+1\}$. The first bound reflects the number of faces in a triangulation, whereas the second one follows directly from Euler's formula (1.4.7) under the assumption that the input is non-planar, i.e., at least one edge has to be removed. Let A denote the bidirected counterpart of the undirected edges E, i.e., for each undirected edge $uv \in E$, the set A contains two directed arcs uv and vu. We add the following binary

variables:

• $x_i \ \forall i \in [\bar{f}]$	Has value 1 if and only if face i exists.
• $c_a^i \ \forall a \in A, i \in [\bar{f}]$	Has value 1 if and only if arc a bounds face i: traversing i in clockwise order visits $\mathrm{e}(a)$ in the orientation of a.
• $p_{u,w}^v \ \forall v \in V, \ u, w \in \mathrm{N}(v)$	Has value 1 if and only if w is the direct successor of u in the cyclic order around v.

To improve comprehensibility, we define the following short-hand notations:

$$p^v(U \times W) \coloneqq \textstyle\sum_{u \in U} \sum_{w \in W} p_{u,w}^v, \qquad x(I) \coloneqq \textstyle\sum_{i \in I} x_i,$$
$$\bar{s}_v(W) \coloneqq \deg(v) - \textstyle\sum_{w \in W} s_{vw}, \qquad c^I(J) \coloneqq \textstyle\sum_{i \in I} \sum_{j \in J} c_j^i.$$

We then complete the facial walk model by the constraints below:

$$n + x([\bar{f}]) = 2 + \textstyle\sum_{e \in E} s_e \tag{2.4a}$$
$$x_i = 1 \qquad \forall i \in [3] \qquad \star \tag{2.4b}$$
$$x_i \geq x_{i+1} \qquad \forall i \in [\bar{f} - 1] \qquad \star \tag{2.4c}$$
$$x_i \leq c^{\{i\}}(A)/3 \qquad \forall i \in [\bar{f}] \tag{2.4d}$$
$$c_a^i \leq x_i \qquad \forall a \in A, i \in [\bar{f}] \tag{2.4e}$$
$$c^{[\bar{f}]}(a) = 1 - s_{\mathrm{e}(a)} \qquad \forall a \in A \tag{2.4f}$$
$$c^{\{i\}}(\delta^-(v)) = c^{\{i\}}(\delta^+(v)) \qquad \forall i \in [\bar{f}], v \in V \tag{2.4g}$$
$$c_{vw}^i \geq c_{uv}^i + p_{u,w}^v - 1 \qquad \forall i \in [\bar{f}], v \in V, u, w \in \mathrm{N}(v) \tag{2.4h}$$
$$c_{uv}^i \geq c_{vw}^i + p_{u,w}^v - 1 \qquad \forall i \in [\bar{f}], v \in V, u, w \in \mathrm{N}(v) \tag{2.4i}$$
$$p^v(u \times \mathrm{N}(v)) = 1 - s_{vu} \qquad \forall vu \in A \tag{2.4j}$$
$$p^v(\mathrm{N}(v) \times w) = 1 - s_{vw} \qquad \forall vw \in A \tag{2.4k}$$
$$p^v(U \times \mathrm{N}(v) \backslash U) \geq \bar{s}_v(\{u, \tilde{u}\}) - 1 \qquad \begin{array}{l} \forall v \in V, \varnothing \neq U \subsetneq \mathrm{N}(v), \\ u \in U, \tilde{u} \in \mathrm{N}(v) \setminus U \end{array} \tag{2.4l}$$

Inequality (2.4a) ensures that the number of nodes, faces, and edges satisfy Euler's polyhedron formula. Constraints (2.4d) account for the fact that each face needs at least three arcs. Conversely, for any arc to be assigned to a face, the face needs to exists (→2.4e). For any arc whose edge is in the planar subgraph there must exist exactly one face that contains the arc (→2.4f). Constraints (2.4g) ensure that the number of inbound arcs equals the number of outbound arcs at a fixed node in a fixed face. By adding

constraints (2.4h,2.4i), we make sure to respect the successor-variables. Constraints (2.4j,2.4k) ensure there are successor variables selected for any edge that is in the solution. The exponentially large set of cut constraints (2.4l) prohibits multiple cycles in the successor relation. Optionally, we can force the use of at least the first 3 faces ($\rightarrow$2.4b), otherwise the solution is outerplanar and thus not maximal; and we can use faces in order of their indices ($\rightarrow$2.4c) to break symmetries.

Special variables/constraints for degree-3 nodes. Consider any degree-3 node v with neighbors u_0^v, u_1^v, u_2^v. If all its incident edges are in the solution, we have two possible cyclic orders. Otherwise, the cyclic order is even unique. Thus, instead of introducing six successor-variables $p^v_{\cdot\cdot\cdot}$ and constraints (2.4h–2.4l), we can use a single binary variable p^v, and straightforwardly simplified constraints, for all $i \in [\bar{f}]$, $j \in [3]$, and all degree-3 nodes v:

$$
\begin{aligned}
c^i_{vu^v_{j+1}} &\geq c^i_{u^v_j v} + (p^v{-}1) - s_{vu^v_{j+1}} \\
c^i_{vu^v_{j+2}} &\geq c^i_{u^v_j v} + (p^v{-}1) - s_{vu^v_{j+2}} + (s_{vu^v_{j+1}}{-}1) \\
c^i_{u^v_j v} &\geq c^i_{vu^v_{j+1}} + (p^v{-}1) - s_{u^v_j v} \\
c^i_{u^v_j v} &\geq c^i_{vu^v_{j+2}} + (p^v{-}1) - s_{u^v_j v} + (s_{vu^v_{j+1}}{-}1) \\
c^i_{vu^v_j} &\geq c^i_{u^v_{j+1} v} - p^v - s_{vu^v_j} \\
c^i_{vu^v_j} &\geq c^i_{u^v_{j+2} v} - p^v - s_{vu^v_j} + (s_{vu^v_{j+1}}{-}1) \\
c^i_{u^v_{j+1} v} &\geq c^i_{vu^v_j} - p^v - s_{u^v_{j+1} v} \\
c^i_{u^v_{j+2} v} &\geq c^i_{vu^v_j} - p^v - s_{vu^v_{j+2}} + (s_{vu^v_{j+1}}{-}1)
\end{aligned}
$$

It can be easily (although tediously) verified by a case analysis that the above inequalities cover every possible configuration of neighbors, where we might assume that there is at least one neighbor since every maximal solution must be connected.

Algorithm engineering & preliminary benchmarks. In our experiments, the special degree-3 node model did not solve more instances but resulted in a marginal reduction (0.8%) of running time; so we use it. The PBS variant on the other hand suffers from the special degree-3 model, solving 9.38% less instances. An ILP variant where we eliminate the solution variables s_e (directly using the containment variables c^i_a instead) solved 3.29% less instances. We refrain from testing polynomially sized models

(betweenness- and index-based instead of constraints (2.4h–2.4l)) as our exact genus experiments suggest this does not pay off [Bey+16].

2.4.2 Schnyder Orders

A *partially ordered set* (*poset*) is a pair $P = (S, \prec)$ where $\prec$ is a strict partial order (transitive, irreflexive, binary relation) over the elements of S. Every poset has a *realizer*, i.e., a set $\mathcal{R}$ of total orders (transitive, antisymmetric, total, binary relation) on S whose intersection is $\prec$ [Szp30]. This means that $x \prec y$ if and only if $x <_i y$ for all $<_i \in \mathcal{R}$. The *Dushnik-Miller dimension* $\dim P$ of P is the minimum cardinality over all realizers of P [DM41]. We associate a poset $P_G = (V \cup E, \prec_G)$ to G such that $x \prec_G y$ if and only if $y = \{v, w\} \in E$ and $x \in y$. The dimension of G is defined as the Dushnik-Miller dimension of P_G.

Theorem 2.4.1 (Schnyder's Theorem, 4.1 and 6.2 of [Sch89])**.** *A graph is planar if and only if its dimension is at most three.*

In fact, a graph with dimension 1 (2) is an isolated node (path, respectively). Therefore, we propose a model to check for dimension three. Fortunately, Schnyder provides another, related and favorable, characterization:

Lemma 2.4.2 (Lemma 2.1 of [Sch89])**.** *A graph $G = (V, E)$ has dimension at most d if and only if there exists a set of total orders $<_1, ..., <_d$ on V such that the intersection of $<_1, ..., <_d$ is empty; and for each edge $\{x, y\} \in E$ and each node $z \notin \{x, y\}$ of G, there is at least one order $<_i$ such that $x <_i z$ and $y <_i z$.*

To use this criterion, we add (additionally to s_e) the following binary variables:

• $t^i_{u,v}$ $\forall i \in [3], \forall u, v \in V : u \neq v$	Has value 1 if and only if $u <_i v$.
• $a^i_{e,v}$ $\forall i \in [3], e \in E, v \in V^e$	Can have value 1 only if $u <_i v$ $\forall u \in e$.

We are now able to complete the Schnyder orders ILP by adding:

$$s_e + \sum_{i=0}^{2} a^i_{e,v} \geq 1 \qquad \forall e \in E, v \in V^e \tag{2.5a}$$

$$a^i_{e,v} \leq t^i_{u,v} \qquad \forall i \in [3], e \in E, u \in e, v \in V^e \tag{2.5b}$$

$$\sum_{i=0}^{2} t^i_{u,v} \leq 2 \qquad \forall u, v \in V : u \neq v \quad \star \tag{2.5c}$$

$$t^i_{u,v} + t^i_{v,w} - 1 \leq t^i_{u,w} \qquad \forall i \in [3], \text{p.d. } u, v, w \in V \tag{2.5d}$$

$$t^i_{u,v} + t^i_{v,u} = 1 \qquad \forall i \in [3], u, v \in V : u \neq v \tag{2.5e}$$

Constraints (2.5a) ensure that for any edge in the solution the Schnyder-property for any non-incident node is satisfied by at least one of the three orders. By inequalities (2.5b), we make sure that the second requirement of the Schnyder-property is respected. Transitivity of the total orders is obtained by (2.5d). Finally, we require totality by adding (2.5e).

As Schnyder states [Sch89], we may omit the intersection criterion (2.5c) as this is satisfied by any non-trivial solution. Note that for any two adjacent edges uv, vw in the solution and any $i \in [3]$, we cannot have $a^i_{uv,w} = a^i_{vw,u} = 1$, since the orders induced by the a-variables are conflicting. Hence, we might pick a single triangle $T = \{e_1, e_2, e_3\}$ in the input graph and assign realizing orders to each edge; thereby v_i denotes the node incident to both of $T \setminus \{e_i\}$:

$$\sum_{j \in [3] \setminus \{i\}} a^j_{e_i, v_i} = 0 \qquad \forall i \in [3] \qquad \star \quad (2.5f)$$

Analogously, we might apply the same symmetry breaking constraint to two adjacent edges if the graph is triangle-free. (Then $e_3 \notin E$, we let $i \in [2]$ but retain the subscript at the sum.)

Algorithm engineering & preliminary benchmarks. For the ILP solver, we tested omitting the symmetry breaking constraints (2.5f) (9.12% less solved instances), omitting intersection constraints (2.5c) (0.85% less), manually separating the transitivity constraints (which does not change the overall number of solved instances but increases running time by 4.00%), and using Theorem 2.4.1—the partial order on $(V \cup E)$—instead (leading to a related but different model that we do not describe here), where we solve 39.89% fewer instances.

Using the PBS solver, we obtain similar results for omitting symmetry breaking constraints (9.37% less) and for omitting intersection constraints (0.79% less). In contrast to above, using lazy transitivity constraints leads to 5.24% fewer solved instances. We did not investigate a PBS variant based on Theorem 2.4.1 as the ILP performance was already strikingly underwhelming. We did consider a variant where we use betweenness variables [Cap+11] to describe each of the three total orders. This allows us to omit the a-variables, but it did not yield satisfactory running time already on rather trivial instances.

2.4.3 Left-Right Edge Coloring

A *Trémaux tree* T is a rooted tree in a graph H such that for any *cotree* edge $\{u, v\} \in E^T_H \coloneqq E(H) \setminus E(T)$, we can traverse the nodes of the tree-path between u and v, such that the levels of the nodes (i.e., their distances in T

to the root) are strictly increasing, see [Tar95]. Any DFS-tree, rooted at the start node, is a Trémaux tree. For any edge e we refer to the node closer to the root of T as $\mathring{e}$ and the other one as $\bullet e$ (this is unique by the Trémaux property). Any Trémaux tree T defines a partial order on the nodes: for each edge $e \in E(T)$ we set $\mathring{e} \prec \bullet e$, the partial order is obtained by extending this relationship to its transitive hull.

Definition 2.4.3 (T-alike and T-opposite relations)**.** *We denote the* meet*, i.e., the closest common ancestor, of two nodes u, v in $\prec$ by $u \wedge v$. De Fraysseix and Rosenstiehl [FR85] define binary relations between cotree edges as follows:*

*P1. For any $\alpha, \beta, \gamma \in E_H^T$ such that $\mathring{\gamma} \prec \mathring{\alpha} \preceq \mathring{\beta} \prec \bullet\alpha \wedge \bullet\beta \wedge \bullet\gamma \prec \bullet\alpha \wedge \bullet\beta$, α and β are T-*alike*.*

*P2. For any $\alpha, \beta, \gamma \in E_H^T$ such that $\mathring{\gamma} \prec \mathring{\alpha} \prec \mathring{\beta} \prec \bullet\alpha \wedge \bullet\beta \wedge \bullet\gamma \prec \bullet\beta \wedge \bullet\gamma$, α and β are T-*opposite*.*

*P3. For any $\alpha, \beta, \gamma, \delta \in E_H^T$ such that $\mathring{\gamma} = \mathring{\delta} \prec \mathring{\alpha} = \mathring{\beta} \prec \bullet\alpha \wedge \bullet\beta \prec \bullet\alpha \wedge \bullet\gamma$, and $\bullet\alpha \wedge \bullet\beta \prec \bullet\beta \wedge \bullet\gamma$, α and β are T-*opposite*.*

Theorem 2.4.4 (Section 2 of [FR85])**.** *A connected graph H with a Trémaux tree T is planar if and only if there exists a partition of E_H^T into two classes, such that any two edges which are T-alike (T-opposite) belong to the same class (different classes, respectively).*

Using this characterization, we design a model that describes a Trémaux tree with a feasible bicoloring of cotree edges for any connected, planar subgraph. We introduce the following set of binary variables, additionally to s_e for all $e \in E$:

• $t_d \ \forall d \in A$	Has value 1 if and only if arc d is in the Trémaux tree T.
• $\ell_{uv} \ \forall u, v \in V$	Has value 1 if and only if node u lies on the path from the root to v in T. Always true for $u = v$ and whenever u is the root of T. Models the partial Trémaux ordering $u \prec v \iff \ell_{uv} = 1$.
• $r_e \ \forall e \in E$	Has value 1 if and only if edge e is colored red (otherwise, $r_e = 0$ and e is colored blue).

Given an arc a, we may directly write r_a when referring to $r_{e(a)}$.

First, we establish a Trémaux tree. It has $n - 1$ edges ($\rightarrow$2.6a), chosen from the planar subgraph ($\rightarrow$2.6b). Its edges seed the partial order on the

nodes (→2.6c). To make sure the order described by the ℓ-variables is exactly the transitive hull of the tree, we need that nodes with the same parent in the tree are not comparable (→2.6d). Whenever two nodes u, v are smaller than a third one, u must be comparable to v (→2.6e). Constraints (2.6f), (2.6h), and (2.6g) model transitivity, reflexity, and antisymmetry, respectively. Finally, the Trémaux tree property—any edge of the planar solution being incident with two comparable nodes—is enforced by constraints (2.6i). Note that the t-variables will always describe a tree, i.e., there are no cycles as this would conflict with the induced partial order by (2.6c,2.6f,2.6g).

$$\sum_{d\in A} t_d = |V| - 1 \tag{2.6a}$$
$$s_{\mathrm{e}(d)} + t_d \leq 1 \qquad \forall d \in A \tag{2.6b}$$
$$t_d \leq \ell_d \qquad \forall d \in A \tag{2.6c}$$
$$\ell_{vw} + \ell_{wv} + t_{uv} + t_{uw} \leq 2 \qquad \forall u \in V, \{uv, uw\} \in A^{\{2\}} \tag{2.6d}$$
$$1 + \ell_{uv} + \ell_{vu} \geq \ell_{uw} + \ell_{vw} \qquad \forall (u, v, w) \in V^{(3)} \tag{2.6e}$$
$$\ell_{uv} + \ell_{vw} \leq \ell_{uw} + 1 \qquad \forall (u, v, w) \in V^{(3)} \tag{2.6f}$$
$$\ell_{uv} + \ell_{vu} \leq 1 \qquad \forall \{u, v\} \in V^{\{2\}} \tag{2.6g}$$
$$\ell_{vv} = 1 \qquad \forall v \in V \tag{2.6h}$$
$$s_e + \ell_{\overset{\bullet}{e}} + \ell_{\overset{\circ}{e}} \geq 1 \qquad \forall e \in E \tag{2.6i}$$

Aiming at cutting off some symmetrical solutions, we may demand that tree edges and deleted edges are colored blue:

$$t_{\overset{\circ}{e}\overset{\bullet}{e}} + t_{\overset{\bullet}{e}\overset{\circ}{e}} + r_e \leq 1 \qquad \forall e \in E \qquad \star \tag{2.6j}$$
$$r_e + s_e \leq 1 \qquad \forall e \in E \qquad \star \tag{2.6k}$$

We may also enforce a unique Trémaux tree for each given assignment of s-variables: pick an arbitrary root node $r \in V$, set its incoming arcs to 0 and those of every other node to 1 (→2.6l,2.6m). Let $<_\pi$ denote a fixed non-cyclic order on the adjacency entries for each node. We may demand that the first feasible edge in this order is always picked for the tree, thus obtaining a distinct feasible DFS-tree for each assignment of s-variables (→2.6n).

$$\sum_{wr\in A} t_{wv} = 0 \qquad \star \tag{2.6l}$$
$$\sum_{wv\in A} t_{wv} = 1 \qquad \forall v \in V \setminus \{r\} \qquad \star \tag{2.6m}$$
$$t_{uw} + \ell_{wv} - s_{uv} \leq 1 \qquad \forall uv <_\pi uw \in A \qquad \star \tag{2.6n}$$

We now establish a feasible bicoloring of the cotree edges. For readability, let $R^{u,v}_{\alpha,\beta,\gamma} \coloneqq C_{\{\alpha,\beta,\gamma\}} + \ell_{\overset{\circ}{\gamma}\overset{\circ}{\alpha}} + \ell_{uv} - 2$, where $C_F \coloneqq \sum_{d\in F} \left(\ell_d - s_{\mathrm{e}(d)} - t_d - t_{\mathrm{rev}(d)} - 1\right)$

for any $F \subseteq A$.

$$\begin{aligned}
P^1_{\alpha,\beta}(\gamma,u,v) &\coloneqq R^{u,v}_{\alpha,\beta,\gamma} + \ell_{\mathring{\alpha}\mathring{\beta}} + \ell_{\mathring{\beta}u} + \ell_{u\mathring{\gamma}} - \ell_{v\mathring{\gamma}} + \ell_{v\mathring{\alpha}} + \ell_{v\mathring{\beta}} - 5,\\
P^2_{\alpha,\beta}(\gamma,u,v) &\coloneqq R^{u,v}_{\alpha,\beta,\gamma} + \ell_{\mathring{\alpha}\mathring{\beta}} + \ell_{\mathring{\beta}u} + \ell_{u\mathring{\alpha}} - \ell_{v\mathring{\alpha}} + \ell_{v\mathring{\beta}} + \ell_{v\mathring{\gamma}} - 5,\\
P^3_{\alpha,\beta}(\gamma,\delta,u,v,w) &\coloneqq R^{u,v}_{\alpha,\beta,\gamma} + C_{\{\delta\}} + \ell_{\mathring{\alpha}u} + \ell_{uv} + \ell_{uw} + \ell_{u\mathring{\alpha}} + \ell_{u\mathring{\beta}}\\
&\quad + \ell_{v\mathring{\alpha}} - \ell_{v\mathring{\beta}} + \ell_{v\mathring{\gamma}} - \ell_{v\mathring{\delta}} - \ell_{w\mathring{\alpha}} + \ell_{w\mathring{\beta}} - \ell_{w\mathring{\gamma}} + \ell_{w\mathring{\delta}} - 9.
\end{aligned}$$

We model coloring restrictions of type P1 (T-alike), P2 (T-opposite by one other cotree edge), and P3 (T-opposite by two other cotree edges) by constraints (2.6o–2.6q), respectively:

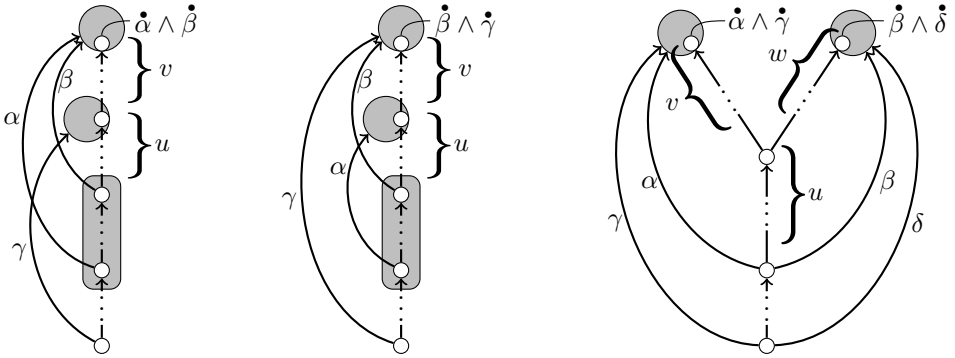
$$\begin{aligned}
&r_\alpha - r_\beta \geq P^1_{\alpha,\beta}(\gamma,u,v)\\
&r_\beta - r_\alpha \geq P^1_{\alpha,\beta}(\gamma,u,v)
\end{aligned}
\qquad
\begin{aligned}
&\forall \alpha,\beta,\gamma \in A \text{ of p.d. edges,}\\
&u \neq v \in V\colon \mathring{\gamma} \neq \mathring{\alpha} \wedge \mathring{\beta} \neq u
\end{aligned}
\tag{2.6o}$$

$$\begin{aligned}
&r_\alpha + r_\beta \geq 1 + P^2_{\alpha,\beta}(\gamma,u,v)\\
&r_\alpha + r_\beta \leq 1 - P^2_{\alpha,\beta}(\gamma,u,v)
\end{aligned}
\qquad
\begin{aligned}
&\forall \alpha,\beta,\gamma \in A \text{ of p.d. edges,}\\
&u \neq v \in V\colon \mathring{\gamma} \neq \mathring{\alpha} \neq \mathring{\beta} \neq u
\end{aligned}
\tag{2.6p}$$

$$\begin{aligned}
&r_\alpha + r_\beta \geq 1 + P^3_{\alpha,\beta}(\gamma,\delta,u,v,w)\\
&r_\alpha + r_\beta \leq 1 - P^3_{\alpha,\beta}(\gamma,\delta,u,v,w)
\end{aligned}
\qquad
\begin{aligned}
&\forall \alpha,\beta,\gamma,\delta \in A \text{ of p.d. edges}\\
&u,v,w \in V\colon v \neq w \text{ and}\\
&\mathring{\alpha} = \mathring{\beta} \wedge \mathring{\gamma} = \mathring{\delta} \wedge \mathring{\gamma} \neq \mathring{\alpha} \neq u
\end{aligned}
\tag{2.6q}$$

To comprehend the latter three constraint classes (2.6o–2.6q), one first needs to understand that for any $F \subseteq A$ we have $-C_F \in \mathbb{N}$ by definition (for any feasible variable assignment) and $C_F = 0$ if and only if each arc of F is a cotree edge of the subgraph induced by the s-variables and directed from the smaller to the larger node. Following this pattern, we define the terms $P^1_{\alpha,\beta}(\gamma,u,v)$, $P^2_{\alpha,\beta}(\gamma,u,v)$, $P^3_{\alpha,\beta}(\gamma,\delta,u,v,w)$ each equal to 0 if and only if we have a configuration of type P1, P2, or P3, respectively, and smaller than or equal to -1 otherwise. Using these terms we can enforce T-alike- and T-oppositeness for any pair α,β as given by constraints (2.6o–2.6q); see Fig. 2.10 for the selection of nodes u, v, and w.

DFS-based branching rule. Apart from a traditional automatic selection of branching variables by the ILP solver, we consider a more specialized scheme. Given a node in the B&B tree, we traverse the locally non-deleted edges of G (i.e., the edges that have a local upper bound of 1) in their unique DFS order until we find an edge e that is not yet chosen to be in the DFS-tree (i.e., the lower bound of the respective arc variable is not 1). We spawn



(a) type P1 (b) type P2 (c) type P3

Figure 2.10: Schematics of configurations inducing T-alike and T-opposite relations with ranges to pick nodes u, v, w from, such that constraints (2.6o–2.6q) are tight. Nodes at the bottom are (closest to) the root. Tree paths are straight (cotree edges are curved), partially dotted lines. Subgraphs of arbitrary structure (possibly just a single node) are shaded in gray.

two new B&B subproblems, where e either is deleted or in the DFS-tree, respectively. While we lose the potential benefit of always branching on a strongly fractional value, we can fix two instead of just one free variable in both new B&B branches.

Algorithm engineering & preliminary benchmarks. Since we cannot hope to explicitly write down all coloring constraints (2.6o–2.6q), we separate on integral ILP solutions and use lazy, i.e., iteratively added, constraints in the PBS variant. We use a simple $\mathcal{O}(|V(G)|^4)$ routine that identifies all violated bicoloring constraints for any integral variable assignment: We iterate over all triples (for type P1 and P2), and all quadruples (for type P3) of cotree edges. For any such triple (resp. quadruple) we test in $\mathcal{O}(1)$ whether the induced constraints are violated. By adding the trivial Euler bound, we may assume that $|E(H)| \in \mathcal{O}(|V(G)|)$. Note that w.l.o.g. we may start the search at edge γ and restrict it to $\alpha, \beta, \delta \in \{\sigma \in E_H^T \mid \mathring{\sigma} \succeq \mathring{\gamma}\}$ and take required adjacencies into account. Also, we may terminate the routine after identifying a large set of violated constraints.

We evaluated ILP variants where we omitted the symmetry breaking constraints (2.6l–2.6n) (49.91% less solved instances), use our custom branch rule while limiting its application to B&B-depth at most 6 (13.22% more) as well as without this limit (32.40% more), and increased the limit of added

constraints per LP run from the default of 100 to 1000 (0.93% more).

Using the PBS solver, we obtain similar results when omitting symmetry breaking constraints (65.74% less). Furthermore, we investigated a separation routine based on directed cuts[5] to cut off infeasible t-variable assignments; this does not seem to be beneficial.

2.4.4 Experimental Evaluation

Setup. All our programs are implemented in `C++`, compiled with GCC 6.3.0, and use the OGDF (version based on snapshot 2017-07-23) [Chi+13]. We use SCIP 4.0.1 for solving ILPs with CPLEX 12.7.1 as the underlying LP solver [Gle+18]. For PBS-based algorithms, we utilize Clasp 3.3.3 [Geb+11]. Each MPS-computation uses a single physical core of a Xeon Gold 6134 CPU (3.2 GHz) with a memory speed of 2666 MHz. We apply a time limit of 20 minutes and a memory limit of 8 GB per computation. Our instances and results, giving running time and skewness (if solved), are available for download at `http://tcs.uos.de/research/mps`.

Instances and configurations. As presented in Section 1.7.1, we use the real-world instance sets North, Rome, and a subset of SteinLib. From SteinLib, we use test-sets `B`, `WRP3`, and `WRP4`, limited to graphs on at most 200 nodes. We also ran benchmarks on sets `I080` and `PUC` but no algorithm was able to solve any such graph. Hence, we omit the respective sets from the experimental comparison. In addition, we use Regular. In [Por08] a set of 46 graphs is considered to evaluate the performance of MPS heuristics. From this set, we use the six graphs `cimi-g1` to `cimi-g6` that were initially collected by Cimikowski [Cim95a]. These graphs have practical relevance or originate from other publications. The latter three (`cimi-g4`–`cimi-g6`) stem from [JTS89], [Kan92], and [TDB88], respectively. We also tried solving other graphs from the set presented in [Por08] but succeeded only on graphs that allow triangulations (i.e., when obtaining trivially optimal solutions). Furthermore, we generated (see Algorithm 3) a set of 600 random graphs that have bounded skewness: we generated 50 graphs for each parameterization $(|V(G)|, d, b) \in \{50, 70\} \times \{1.25, 1.50\} \times \{5, 10, 15\}$, where d denotes the density of a random planar graph and b is the number of random edges added to this graph (hence an upper bound on the skewness). We denote this set of instances by SkewBound.

For formulations that allow multiple configurations, we determined the

[5] Our constraints take the form $\sum_{w \in W, v \in V \setminus W} t_{wv} \geq 1$ for all W with $\{r\} \subseteq W \subsetneq V$.

ALGORITHM 3: Generates a random graph with bounded skewness.

Input: desired number of nodes n, density d, and bound b on skewness

$G \leftarrow K_1$
while $|V(G)| < n$ **do**
  `// generate a random tree on` n `nodes`
  $v \leftarrow$ node of G, chosen uniformly at random
  $w \leftarrow$ new node
  $G \leftarrow G + w + vw$
$L \leftarrow$ empty queue
L.enqueue(outer face of G)
while $|E(G)|/|V(G)| < d$ **do**
  `// add new edges while maintaining planarity`
  $f \leftarrow L$.dequeue()
  $e \leftarrow$ edge $\notin E(G)$ that splits f, chosen uniformly at random
  $G \leftarrow G + e$
  $F \leftarrow$ set of faces incident with e in G
  foreach $g \in F$ **do**
    L.enqueue(g) unless g is a triangle
foreach $i \in \{1, ..., b\}$ **do**
  `// add new edges to make` G `(possibly) non-planar`
  $e \leftarrow$ edge $\notin E(G)$, chosen uniformly at random
  $G \leftarrow G + e$
reject G if it is planar

Table 2.3: Percental ratios of solved instances. The Kuratowski ILP dominates all other algorithms.

	Rome	North	SteinLib	Regular	SkewBound
# instances	8249	423	31	380	600
ILP Algorithms					
Kuratowski	85.70	73.75	67.74	34.74	99.67
Facial Walks	17.82	29.78	12.90	6.58	16.50
Schnyder Orders	21.69	48.22	12.90	13.68	21.00
L.-R. Coloring	36.64	60.75	12.90	19.74	35.83
PBS Algorithms					
Kuratowski	77.43	69.73	48.39	15.79	89.83
Facial Walks	15.21	30.02	6.45	1.05	5.00
Schnyder Orders	46.24	61.93	22.58	10.53	21.00
L.-R. Coloring	65.07	66.43	32.26	15.26	74.83

most promising one in a preliminary benchmark on a set of 1 224 graphs from North and Rome, as already mentioned above. This *fixed* subset of instances was sampled by partitioning the instances into buckets based on the number of nodes and choosing a fixed number of graphs from each bucket with uniform probability. For parameters where we had a non-binary choice (e.g., heap size in ILP separation) we rely mostly on the values identified in [Hed17].

Our algorithms use strong primal heuristics, whose common foundation is a maximal planar subgraph algorithm based on the simpler cactus algorithm by Călinescu et al., with approximation ratio 7/18, that was identified to be among the practically best heuristics, cf. Section 2.2.

Results. Table 2.3 summarizes the ratios of solved instances. Evidently, the Kuratowski ILP dominates all implementations. To our surprise, the rather intricate left-right edge coloring model constitutes the most successful one among the new variants. The facial walk model falls behind all other formulations. A similar picture is obtained from a more detailed look at the success rates (cf. Figs. 2.12 and 2.13).

In Table 2.4 we see that the instance set from [Cim95a] is not hard: Its skewness is at most 4 and all PBS -based models but Facial Walks solve

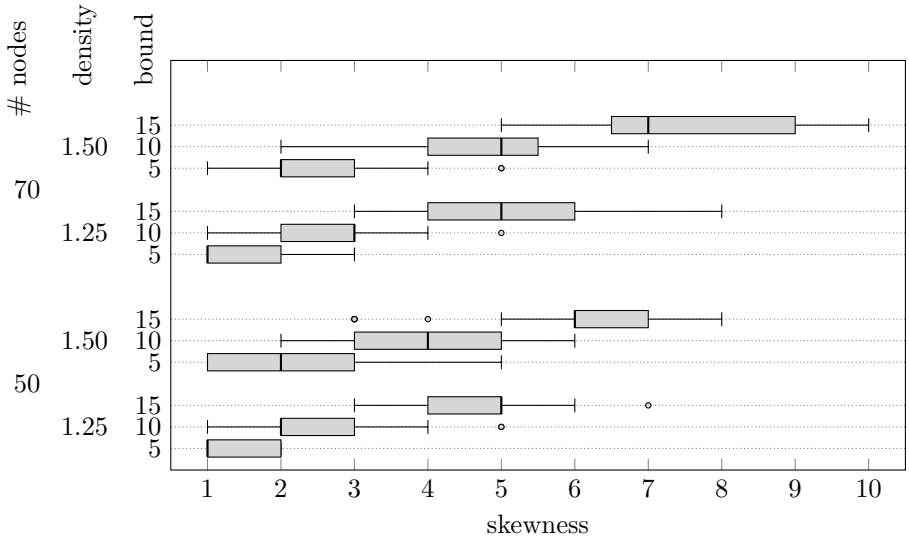


Figure 2.11: Skewness of random generated instance set SkewBound.

Table 2.4: Obtained dual bounds on MPS for graphs by Cimikowski. We omit the prefix "`cimi-`" from each instance's name (e.g., `g1` instead of `cimi-g1`). The variant PBS Facial Walks did not provide any dual bounds.

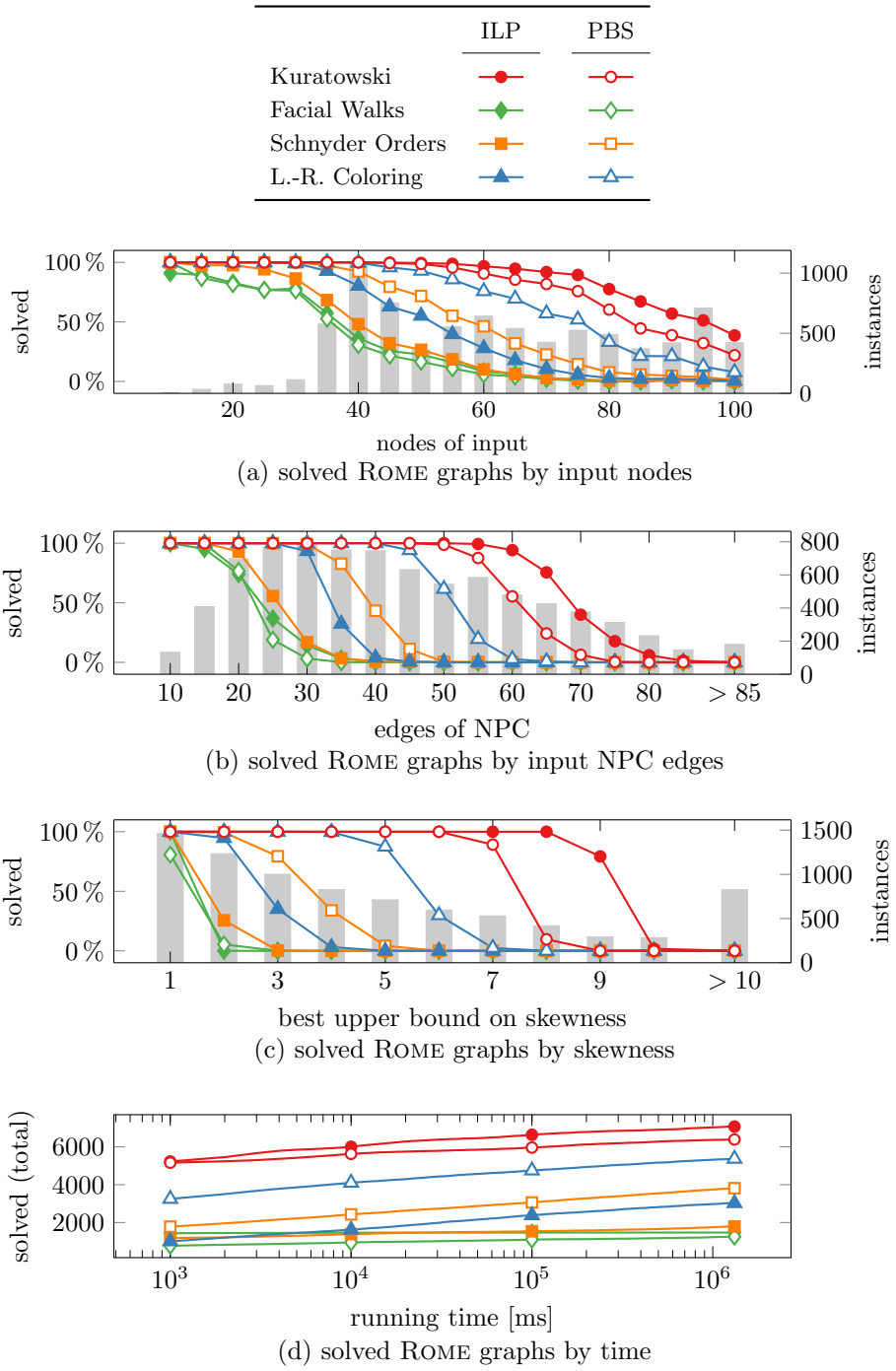
	`g1`	`g2`	`g3`	`g4`	`g5`	`g6`
# nodes	10	60	28	10	45	43
skewness	2	1	2	2	3	4
edges in MPS	19	165	73	20	82	59
ILP Algorithms						
Kuratowski	19	165	73	20	82	59
Facial Walks	20	165	74	21	83	62
Schnyder Orders	19	165	74	20	82	62
L.-R. Coloring	19	166	75	20	83	63
PBS Algorithms						
Kuratowski	19	165	73	20	82	59
Schnyder Orders	19	165	73	20	82	59
L.-R. Coloring	19	165	73	20	82	59

Table 2.5: Percental ratio of solved SteinLib instances. The most successful algorithms are marked in bold.

SteinLib subset	B	WRP3	WRP4
# instances	18	8	5
ILP Algorithms			
Kuratowski	**50**	**88**	**100**
Facial Walks	17	0	20
Schnyder Orders	17	0	20
L.-R. Coloring	17	0	20
PBS Algorithms			
Kuratowski	**50**	38	60
Facial Walks	6	0	20
Schnyder Orders	28	13	20
L.-R. Coloring	39	13	40

Table 2.6: Number of solved instances among Regular (20 graphs for each parameterization).

# nodes	10		20			30		50		100
node degree	4	6	4	6	10	4–10	20	4–20	40	4–40
ILP Algorithms										
Kuratowski	20	20	20	0	20	0	20	0	20	0
Facial Walks	8	17	0	0	0	0	0	0	0	0
Schnyder Orders	20	18	0	0	11	0	0	0	3	0
L.-R. Coloring	20	20	0	0	20	0	12	0	3	0
PBS Algorithms										
Kuratowski	20	20	20	0	0	0	0	0	0	0
Facial Walks	4	0	0	0	0	0	0	0	0	0
Schnyder Orders	20	20	0	0	0	0	0	0	0	0
L.-R. Coloring	20	20	18	0	0	0	0	0	0	0



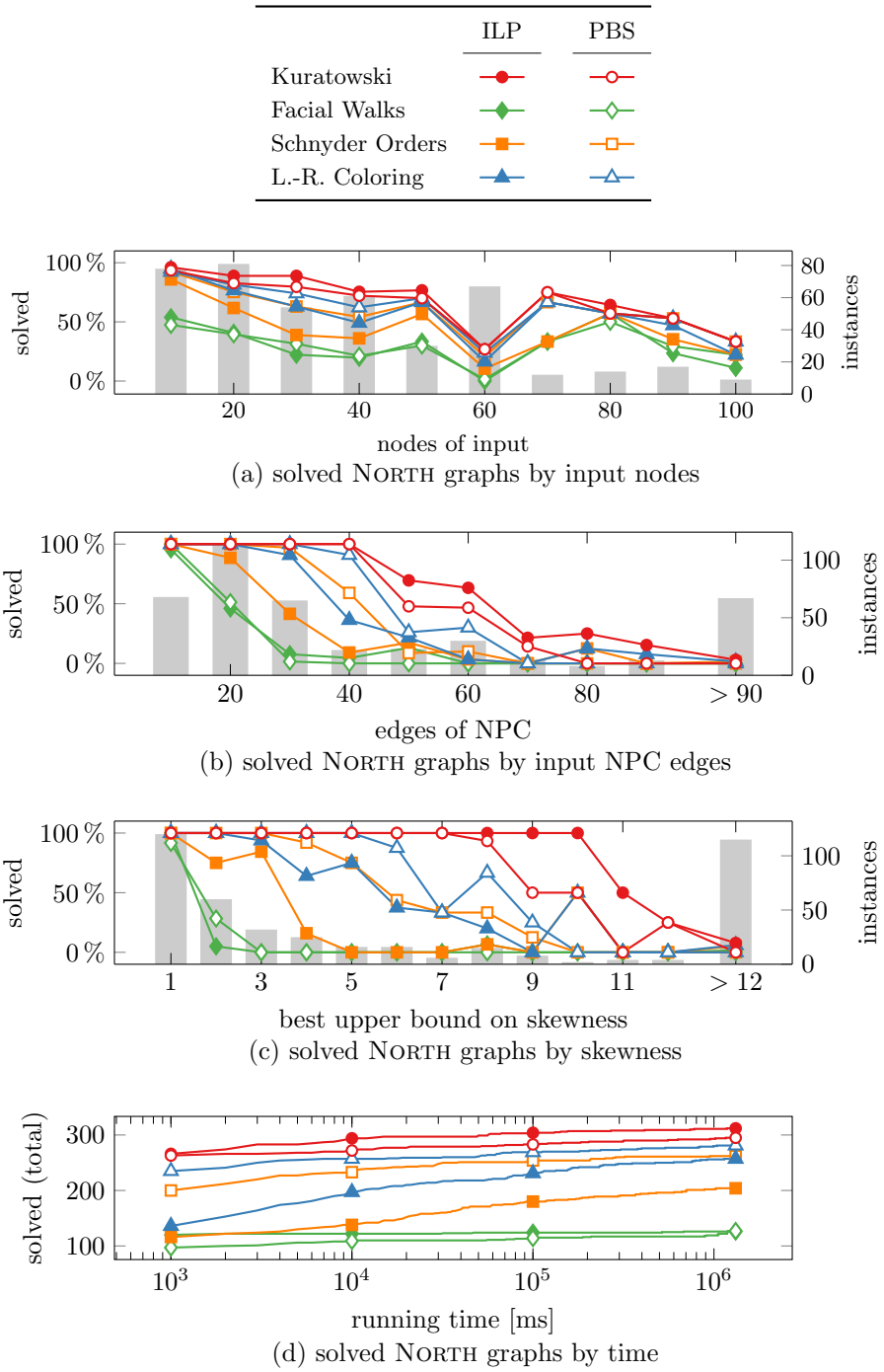
(a) solved Rome graphs by input nodes

(b) solved Rome graphs by input NPC edges

(c) solved Rome graphs by skewness

(d) solved Rome graphs by time

Figure 2.12: Success rate and running time on Rome graphs.



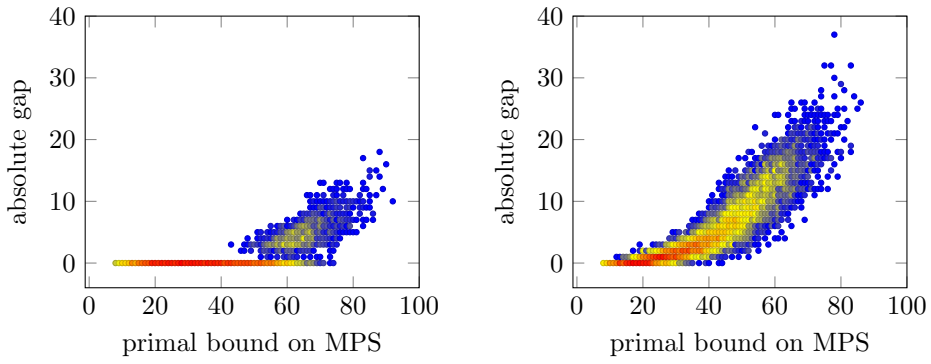
(a) solved NORTH graphs by input nodes

(b) solved NORTH graphs by input NPC edges

(c) solved NORTH graphs by skewness

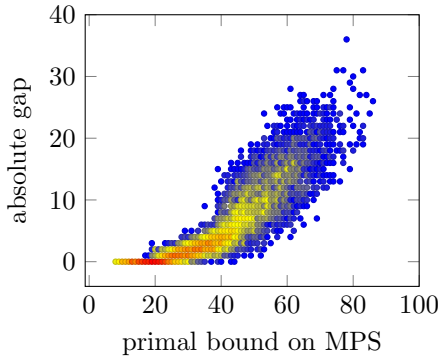
(d) solved NORTH graphs by time

Figure 2.13: Success rate and running time on NORTH graphs.

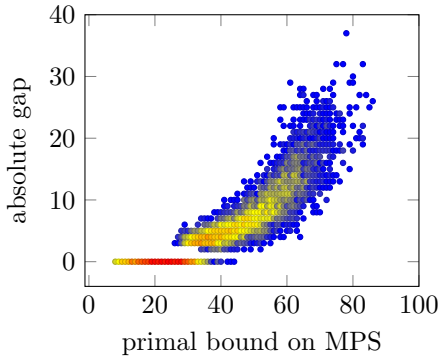


(a) Kuratowski

(b) Facial Walks

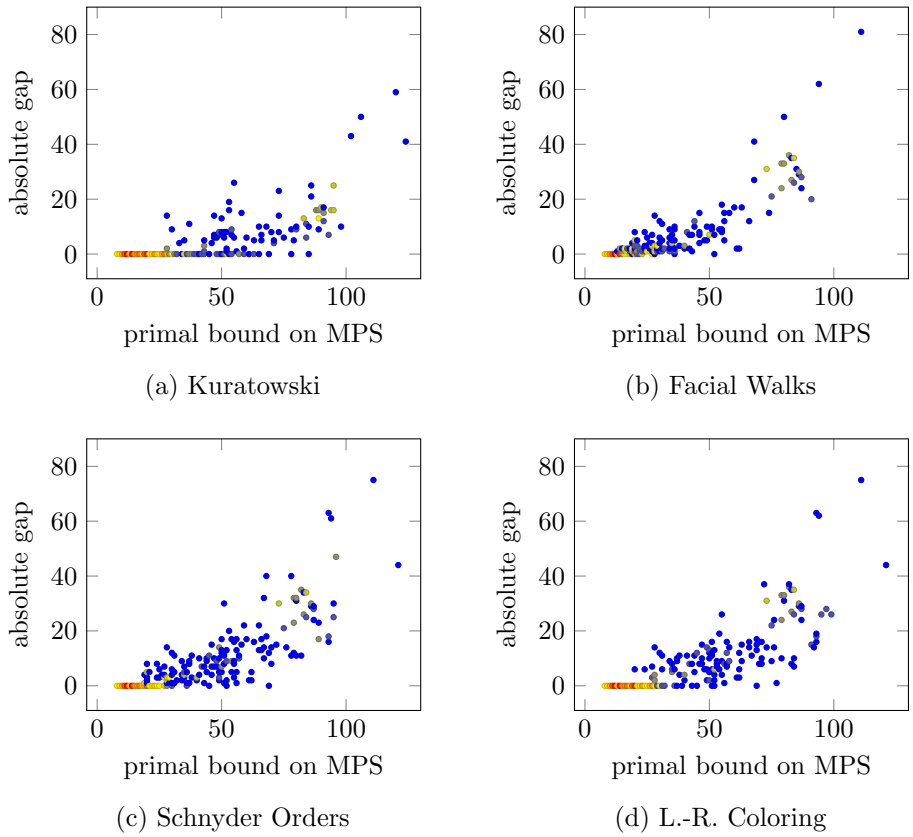


(c) Schnyder Orders



(d) L.-R. Coloring

Figure 2.14: Comparison of ILP primal and dual bounds on ROME graphs. Reddish colors correspond to many instances represented by the same dot. Single instances are blue. Absolute gap refers to dual bound minus primal bound.



(a) Kuratowski (b) Facial Walks

(c) Schnyder Orders (d) L.-R. Coloring

Figure 2.15: Comparison of ILP primal and dual bounds on NORTH graphs. Reddish colors correspond to many instances represented by the same dot. Single instances are blue. Absolute gap refers to dual bound minus primal bound.

Table 2.7: Number of solved instances among the random bounded graphs (50 graphs for each parameterization).

# nodes	50						70					
density	1.25			1.5			1.25			1.5		
upper bound	5	10	15	5	10	15	5	10	15	5	10	15
ILP Algorithms												
Kuratowski	50	50	50	50	50	49	50	50	50	50	50	49
Facial Walks	36	8	0	13	0	0	27	4	0	11	0	0
Schnyder Orders	44	13	0	22	0	0	31	5	0	11	0	0
L.-R. Coloring	50	34	2	41	3	0	48	17	0	20	0	0
PBS Algorithms												
Kuratowski	50	46	38	49	50	41	49	46	46	50	50	24
Facial Walks	16	0	0	1	0	0	9	1	0	2	0	0
Schnyder Orders	50	43	17	49	39	8	49	41	9	48	23	0
L.-R. Coloring	50	45	37	47	49	10	48	46	29	50	35	0

each graph to optimality. This highlights the weak performance of the Facial Walks approach, independently of the used solver. On the contrary, a large performance gap between the two solving methods can be observed for the left-right coloring approach. Table 2.5 confirms the strength of the Kuratowski approach within each subset of the SteinLib. Table 2.6 lists the number of solved instances among all parameterizations of expander graphs. It indicates that all expanders that were solved by one of the new algorithms were also solved by the respective Kuratowski variant. One also observes that more edges do not necessarily lead to harder instances: The Kuratowski model often solves only dense enough instances, typically allowing solutions that are close to a triangulation. Similarly, Table 2.7 shows the number of solved instances among the parameterizations of the random skewness-bounded graphs. Once again, the Kuratowski ILP not only dominates all other implementations but is also able to solve all but two graphs. We observe that the upper bound given by the number of potentially non-planar edge insertion steps is often rather loose, cf. Fig. 2.11.

In Fig. 2.12a (Fig. 2.12b), we show the relative number of solved instances among the Rome graphs over the nodes in the input (resp. number of edges in the non-planar core), clustered to the nearest multiple of five. As expected,

the more edges there are in the core, the harder the instance is in practice. This is particularly clear on the ROME graphs and becomes a little distorted on the NORTH graphs, see Figs. 2.13a and 2.13b, that include some instances where we have to delete very few edges to obtain a (near) triangulation with an (almost) trivial upper bound. Figs. 2.12d and 2.13d show the number of solved instances over our total running time. The running time is represented logarithmically. Again, the Kuratowski ILP is the clear winner and solves more instances than any other variant at any point in time. While the number of solved instances for all algorithms is already very substantial after the first milliseconds and only very slowly increases over the course of 20 minutes, we can see that some algorithms gain more than others from an increase in running time. Surprisingly, the Schnyder orders ILP seems to benefit only on the considerably harder NORTH graphs from increasing the running time. In most cases, particularly on the ROME and NORTH instances, the PBS variants are stronger than their ILP counterparts, with a clear exception for the Kuratowski model. We assume that this effect is due to the weakness of the respective LP relaxations, which fail to give strong enough bounds. Thus, the much faster enumeration strategies of PBS solvers win, not spending time on comparably futile LP computations. Finally, Figs. 2.12c and 2.13c relate upper bounds on the skewness with the number of solved instances. We can see that there is little success on graphs with a skewness larger than 12, on both the ROME and NORTH set. The same holds, although not as clear cut, for the other instance sets. Figs. 2.14 and 2.15 show a large gap for many instances when using the ILP approach. Evidently, only the Kuratowski ILP reaches a gap below 20 on all ROME graphs. This is in stark contrast to all other ILPs where we obtain gaps that are twice as large. For example, the topmost outlier `grafo8882.100.lgr` with gap 37 for Facial Walks and L.-R. Coloring and gap 36 for Schnyder Orders, has a gap of only 18 in the Kuratowski model. Interestingly, although the L.-R. Coloring model is able to optimally solve many ROME and NORTH graphs, there are no instances that result in a gap of exactly one in either of the two instance sets, we have no good explanation for this behavior. We further observe that even for the supposedly simple ROME graphs, there are instances that do not seem to be solvable by the strongest choice, i.e., the Kuratowski model. Considering the large gaps that can be observed, even a generous relaxation of the running time limit is likely not sufficient to solve these graphs with the suggested models.

Conclusion. The main goal in this section was to investigate novel ways of approaching the MPS problem, after over two decades of no progress w.r.t. exact models. We succeeded in the sense that we showed that there are indeed viable alternatives. However, we also showed experimentally that a modern implementation of the old Kuratowski formulation remains the strongest option to solve MPS in practice. Although negative, this is an interesting observation.

We should keep in mind that the thereby required efficient separation builds upon years of algorithmic development [CMS07; Joh04], and it is the only ILP where we currently know how to (heuristically) separate on *fractional* solutions. Equipped with similar tools, i.e., a sensible rounding scheme and a linear-time separation routine (e.g., a modified left-right planarity test), the left-right edge coloring formulation might yield very competitive performance. This, in fact, may be a reasonable target for future research.

For the genus problem, a facial walk model similar to our MPS formulation is the only known feasible approach. However, we clearly see that it is not favorable for MPS as we have stronger and more direct options at our disposal. The facial walk model optimizes over all possible embeddings (there are exponentially many already for a fixed subgraph) of all planar subgraphs, which might help explain its underwhelming performance. The Schnyder orders model does not perform very well in practice despite its very elegant characterization. This might be due to the fact that in contrast to the left-right edge coloring, we search for three feasible orders on the planar subgraph instead of just one (the partial order corresponding to the Trémaux tree). To solve the Schnyder orders model efficiently, a fast solver for linear ordering problems seems to be required. The Schnyder and left-right edge coloring PBS formulations usually beat their ILP counterparts, indicating that their LP relaxations are rather weak. As expected, the expander graphs constitute a particularly hard class of instances and may be a good starting point for tuning and extending our algorithms.

Finally, the strong performance of the Kuratowski model (in particular the ILP variant) is a clear indication that it deserves more attention in the future. In fact, just recently initial progress was made towards a stronger model founded on the Kuratowski approach [CW18]. In the upcoming section, we turn our attention towards this novel Kuratowski-based approach.

2.5 Stronger ILP Models Based on Small Cycles

Here, we want to discuss potential improvements to the state-of-the-art Kuratowski-based model by Mutzel. To this end, we will investigate several new constraint classes that consider small cycles in the graph.

We use the ILP model by Mutzel as the basis of our extensions (see Section 2.1). As such, we denote it, without any of the below extensions, by 'ε-model'. Let us present new constraints for this planar subgraph polytope (or a lifted version thereof). All but the first class of new constraints require the introduction of new variables based on cycles, leading to the *cycle model*. For each constraint class we first give some motivation and intuition for its feasibility, before discussing its technical details. We then describe—provided the class is large—separation routines that quickly identify violated constraints, and usually show that it strengthens our ILP model.

Such proofs of strength always proceed in the following manner: First, we describe an integrally edge-weighted graph that is used as input. Note that (integral) edge-weights are naturally obtained by contracting bundles in the input and other preprocessing techniques. We restrict ourselves to instances that cannot be reduced by standard techniques [CG09]. Next, we give an LP-feasible solution with objective value OBJ for the model that does not use the new constraints. In particular, we also show that the solution satisfies *all* traditional Kuratowski constraints (2.1). Finally, we show that there is no LP-feasible solution with objective value OBJ when using (a subset of) the new constraints.

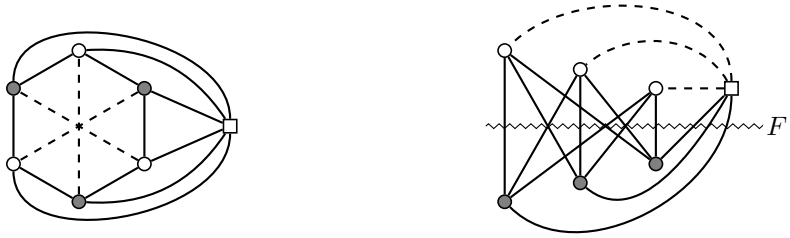
Generalized Euler Constraints. We know from [JM96] that the inequality $|E(G)| \leq 2|V(G)| - 4$ is facet-defining for complete biconnected graphs. We are interested in a class of similar constraints for dense subgraphs with large girth ϕ. The following lemma is folklore:

Lemma 2.5.1. *A planar graph G satisfies*

$$|E(G)| \leq \big(|V(G)| - 2\big)\phi(G)/\big(\phi(G) - 2\big).$$

Proof. Let $n \coloneqq |V(G)|$, $m \coloneqq |E(G)|$, and π denote an embedding of G. For any face of π we require at least $\phi(G)$ half-edges. Thus, the number f of faces in π is bounded by $f \leq 2m/\phi(G)$. Using Euler's formula, we obtain $n - m + (2m/\phi(G)) \leq 2$, the claimed results follows when solving for m. □

We thus derive a feasible *generalized Euler constraint* for any subgraph



(a) Edges of the maximum matching M in G' are dashed.

(b) The $K_{3,4}$-subgraph (dashed edges are removed from $K_{3,3,1}$).

Figure 2.16: The input $K_{3,3,1}$ for showing the strength of the generalized Euler constraints.

$H \subseteq G$:

$$|E(H)| - s\big(E(H)\big) \leq \big(|V(H)| - 2\big)\phi(H)/\big(\phi(H) - 2\big) \quad \forall H \subseteq G \tag{2.7}$$

We note that this bound can sometimes be improved: for constraints (2.7) to be satisfied with equality it is necessary that $V(H) \equiv 2 \pmod{\phi(H) - 2}$ if $\phi(H)$ is odd and $V(H) \equiv 2 \big(\mathrm{mod}\ (\phi(H) - 2)/2\big)$ otherwise [FST18]. However, we did not implement this in our algorithms.

Lemma 2.5.2. *The generalized Euler constraints* (2.7) *strengthen the ε-model.*

Proof. Let G denote the $K_{3,3,1}$, i.e., the complete bipartite graph with three nodes in each partition plus a new node that is connected to each of the other six nodes (see Fig. 2.16).

Consider the ε-model on G. We show the fact for OBJ $= 3/2$. Since $\phi(G) = 3$, the traditional Euler constraint only gives an upper bound of 15 on the number of edges in the planar subgraph, but $|E(G)| = 15$. Let v denote the node in the cardinality-1 partition and M denote a perfect matching in $G' \coloneqq G - v$. For any cardinality-2 subset $S \subset M$, the graph $G - S$ is planar (see Fig. 2.16a). Hence, we satisfy all Kuratowski constraints by setting $s_e = 1/2$ for all $e \in M$. It follows that an LP-value of $3/2$ is feasible without the generalized Euler constraints.

Consider an edge cut F in $K_{3,3,1}$ that partitions the nodes $V(G')$ such that $E(G')$ is fully contained in F, and places v in either of the two partitions. F induces a $K_{3,4}$-subgraph H with $\phi(H) = 4$ (see Fig. 2.16b). The generalized Euler constraint on H is $s(E(H)) \geq 2$, hence improving the dual bound. □

We separate constraints (2.7) heuristically by seeking dense, high-girth subgraphs using two different methods. First, using the current fractional solution, we assign weight $1 - s_e$ to each edge e and approximate a maximum cut [MU05, Section 6.3], obtaining a girth-4 subgraph. If (after postprocessing, see below) this does not yield a violated constraint, we try a second method: We set a target girth μ and iteratively add edges in ascending order of their LP-value to an initially empty graph, while updating the shortest paths between all node pairs. Upon adding an edge e, we check whether e would create a cycle of length less than μ, in which case we discard e instead. We may repeat this process for different values of μ. After each of the above attempts, we apply a postprocessing: Let H denote a girth-μ subgraph. The *contribution* of a node $v \in V(H)$ is defined by $|\delta_H(v)| - \sum_{e \in \delta_H(v)} s_e - \mu/(\mu-2)$. We iteratively remove nodes with negative contribution from H. In particular, this will remove all degree-1 nodes.

2.5.1 Cycle Model

We now want to bound the number of edges in the planar subgraph by the number of its small faces. Even though compelling from a theoretical standpoint, it is infeasible to generate all potential faces of all planar subgraphs of a given graph (already for bounded length). However, we know that traversing the border of any face of a connected subgraph H traverses at least one cycle unless H is a tree. We will relate the number of small faces in any planar subgraph of a graph G to the number of small cycles in G. One may also view this as a way to further generalize Euler constraints: many—in particular sparse—graphs have low girth only due to very few cycles of small length.

We may assume any (maximal) primal solution to be connected and non-outerplanar as it could be trivially improved otherwise. Also observe that we cannot require faces to uniquely map to cycles in general. Consider for example a cycle graph (two faces with the same cycle) or a non-biconnected graph (each cut-node occurs twice in at least one face; cycles contain nodes at most once) Note that there are biconnected graphs that have no biconnected MPS.

Lemma 2.5.3. *For every connected, planar but non-outerplanar subgraph H of G, there exists an embedding of H such that we can assign a unique cycle α to every face f where all edges of α occur on the boundary of f.*

Proof. Let $H \subset G$ be as defined in the claim. There exists some biconnected component $B^\star$ of H that is neither a cycle nor an edge since H is not

outerplanar. Choose an embedding of H and pick some face of $B^\star$ as the outer one. For every biconnected component B that is not just an edge, we iterate over the inner faces of B. Each inner face f of B directly corresponds to a cycle as a biconnected graph contains neither cut-nodes nor bridges. (Observe that an inner face of B might in fact be much larger in H since we ignore other components nested in this face.) Ultimately, we assign the cycle induced by the outer face in H to the (last remaining) outer face. Since $B^\star$ is not a cycle we do not assign any cycle twice. □

We denote the number of faces whose degree satisfies some property $\mathcal{P}$ by $f_{\mathcal{P}}$.

Lemma 2.5.4. *Given a connected, planar graph H on n nodes and m edges, for each embedding of H with exactly $f_{=d}$ faces of degree $d \in \{3, 4, ..., 2m\}$, we have*

$$m = 3n - 6 - \textstyle\sum_{d=3}^{2m}(d-3)f_{=d}. \tag{2.8}$$

Proof. Every face in any embedding of H has degree at least 3 and at most $2m$. For every face f of degree d we can add $d-3$ edges that split f into $d-2$ triangles without violating planarity. After performing this operation for each face, we obtain a planar triangulated graph, i.e., a graph that has exactly $3n-6$ edges. □

Let $\mathcal{C}_d(G)$ denote all cycles of length d in G. We set $D \geq 3$ to the *maximum cycle length* that we want to investigate; this parameter will control the number of additionally generated variables. Let $\mathcal{C}_{\leq D}(G)$ denote the set of cycles with length at most D. For every cycle $\alpha \in \mathcal{C}_{\leq D}(G)$ we generate a variable $c_\alpha \in \{0, 1\}$.[6] We force such a variable to 0 if any edge of the respective cycle is removed and allow at most two cycles per edge in the MPS:

$$\textstyle\sum_{\alpha \in \mathcal{C}_{\leq D}(G)\colon e \in E(\alpha)} c_\alpha \leq 2\,(1 - s_e) \qquad\qquad \forall e \in E(G) \tag{2.9}$$

Note that constraints (2.9) resemble the requirement for each edge to appear in at most two faces (subject to Lemma 2.5.3). We discuss its correctness below. Let $c(d) \coloneqq \sum_{\alpha \in \mathcal{C}_d(G)} c_\alpha$.

[6] Intuitively, we want $c_\alpha = 1$ if and only if α is (part of) a face, see below for details. In terms of correctness, we need not but can actively force these variables to be binary, cf. Section 2.5.3.

Lemma 2.5.5. *For every connected, planar but non-outerplanar subgraph H of G, there exists an embedding π of H and a feasible assignment w.r.t.* (2.9) *of cycle variables such that for each $d \leq D$ the number $f_{=d}$ of faces with degree d in π is bounded from above by*

$$f_{\leq d} \leq \sum_{k=3}^{d} c(k), \quad \textit{or equivalently} \quad f_{=d} \leq \sum_{k=3}^{d} c(k) - f_{<d}.$$

Proof. We assign cycle variables following the proof of Lemma 2.5.3. Hence, there is a unique cycle variable assigned to each face such that the length of the cycle is at most the degree of its face. The variable assignment is feasible since we pick only edges contained in H and pick at most two cycles incident with any such edge. □

Theorem 2.5.6. *For any maximum planar subgraph of a graph G on n nodes and m edges there exists a feasible variable assignment that satisfies* (2.9) *and the* cycle constraint

$$(D-1)\big(m - s(E(G))\big) \leq (D+1)(n-2) + \sum_{d=3}^{D}(D+1-d)c(d). \quad (2.10)$$

Proof. Starting with (2.8) on any connected, planar subgraph of G that has $m - s(E(G))$ edges, we relax the equality by using the same coefficient for all faces of large degree as in

$$m - s(E(G)) \leq 3n - 6 - \sum_{d=3}^{D}(d-3)f_{=d} - (D-2)f_{>D}.$$

By replacing $f_{>D}$ in Euler's formula, $(f_{>D} + f_{\leq D}) + n - \big(m - s(E(G))\big) = 2$, we obtain

$$(D-1)\big(m - s(E(G))\big) \leq (D+1)(n-2) + \sum_{d=3}^{D}(D+1-d)f_{=d}. \quad (2.11)$$

The claimed cycle constraint is finally obtained by applying Lemma 2.5.5 to iteratively replace $f_{=D'}$ for $D' = D, D-1, ..., 4, 3$ by the upper bound (note that $f_{<3} = 0$), as sketched below for the (generalized, iteratively re-appearing) rightmost summand of (2.11):

$$\sum_{d=3}^{D'}(D'+1-d)f_{=d} \leq \sum_{d=3}^{D'-1}\big((D'-1)+1-d\big)f_{=d} + \sum_{d=3}^{D'} c(d) \quad \square$$

Relaxations and D-Hierarchy

We now turn our attention to LP-relaxations of the cycle model. We show that there is a hierarchy of gradually stronger LPs induced by the maximum cycle length D. Let the *cycle model* CM_D consist of the ε-model, the cycle variables for cycle lengths up to D, and the corresponding constraints (2.9),(2.10). If $D = 2$ were allowed, CM_2 would be exactly the ε-model.

Lemma 2.5.7. *For any solution to the relaxation of* CM_D*, it holds that*

$$\sum_{d=3}^{D}(d-2)c(d) \leq 2n-4.$$

Proof. Assume the contrary, i.e., $\sum_{d=3}^{D}(d-2)c(d) > 2n-4$. It follows that $\sum_{d=3}^{D} dc(d) > 2n-4+2\sum_{d=3}^{D} c(d)$ and hence $m-s(E(G)) > n-2+\sum_{d=3}^{D} c(d)$ by the sum of constraints (2.9). Plugging this bound on the number of edges into the cycle constraint (2.10), we obtain $\sum_{d=3}^{D}(d-2)c(d) < 2n-4$, a contradiction. □

It is not immediately clear, that decreasing the maximum cycle length maintains LP-feasibility, as some variables are removed and the cycle constraint is replaced. By employing Lemma 2.5.7, we can show the following fact.

Lemma 2.5.8. *Model* CM_{D+1} *is at least as strong as* CM_D*.*

Proof. Let $A \coloneqq (\bar{s}, \bar{c})$ denote a feasible variable assignment for CM_{D+1}. By eliminating cycle variables of length $D+1$ from A we obtain a feasible variable assignment for CM_D with the same objective value: as we do not change the value of $\bar{s}$, we respect all Kuratowski constraints. The sum of cycle variables for each edge does not increase, thus, constraints (2.9) are respected as well. It remains to show that the cycle constraint is satisfied which is obtained in the following way, starting with the trivial identity:

$$\sum_{d=3}^{D} D(D+1-d)c(d) = \sum_{d=3}^{D} D(D+1-d)c(d)$$

We apply Lemma 2.5.7 for $D+1$.

$$2n-4+\sum_{d=3}^{D} D(D+1-d)c(d) \geq \sum_{d=3}^{D+1}(D-1)(D+2-d)c(d)$$

Adding $(D^2+D)(n-2)$ to both sides and afterwards dividing by $D(D-1)$ gives the right-hand sides of equation (2.15), divided by factor $D-1$ (resp. D).

$$\begin{aligned}&\Big((D+1)(n-2)+\textstyle\sum_{d=3}^{D}(D+1-d)c(d)\Big)\cdot(D-1)^{-1}\\ \geq &\Big((D+2)(n-2)+\textstyle\sum_{d=3}^{D+1}(D+2-d)c(d)\Big)\cdot D^{-1}\end{aligned}$$

The upper bound on the number of edges due to D is no less than that due to $D+1$. □

Lemma 2.5.9. *Model* CM_{D+1} *is stronger than* CM_D*.*

Proof. Consider the complete graph K_k on $k \geq 5$ nodes. Pick any number $\mu \geq D+1$. We subdivide every edge of K_k using $\xi := \lfloor \mu/3 \rfloor$ additional nodes. The resulting graph K_k^μ has girth at least μ, i.e., it has no cycles of length $\leq D$. We observe that $skew(K_n^\mu) = skew(K_k) = k(k-1)/2 - 3k + 6$, independent of μ. We show that increasing the maximum cycle length from D to $D+1$ cuts off all previously optimal LP solution.

Since K_k^μ has girth μ there can be at most $(|V(K_k^\mu)| - 2)\mu/(\mu - 2)$ edges in any planar subgraph. As there are no cycle variables, the cycle constraint (2.10) approaches this value from above for increasing D. Any feasible solution that tightly satisfies the cycle constraint is an optimal one. The Kuratowski constraints (2.1) on the other hand are already satisfied by deleting each edge partially with $s_e = 1/(9\xi)\ \forall e \in E(K_n^\mu)$, since each subdivision requires at least 9ξ edges, still allowing LP-solutions with value $k(k-1)/18$. □

Overall, increasing the maximum cycle length strengthens our LP relaxations (leading to fewer LP-computations), but this comes at the cost of increasing the variable space (leading to slower LP-computations). It is imperative to find a good trade-off between these two.

2.5.2 Strengthening the Cycle Model

We now extend the cycle model further by introducing new constraint classes. Only the first such extension requires yet additional variables.

Pseudo-Tree Extension. Observe that degree-1 nodes in the solution deteriorate the cycle constraint's bound: given a face f that contains a degree-1 node, we can set the variable of a cycle with length at most $\deg(f) - 2$ to 1. We introduce new variables $t_v \in \{0,1\}$ for all $v \in V(G)$ and $t_{vw} \in \{0,1\}$ for all $v, w \in V(G)$ with $\{v,w\} \in E(G)$. They label nodes and directed edges (*arcs*) as *pseudo-trees*: any node with at most one unlabeled neighbor (in particular any degree-1 node) is to be labeled. This can be achieved by:

$$t_{vw} + t_{wv} + s_{\{v,w\}} \leq 1 \qquad \forall \{v,w\} \in E \qquad (2.12)$$

$$\textstyle\sum_{w \in N(v)} t_{vw} \geq t_v \qquad \forall v \in V(G) \qquad (2.13)$$

$$t_v + \deg_G(v) - \textstyle\sum_{w \in N(v)} t_{wv} - \sum_{w \in N(v)} s_{vw} \geq 2 \qquad \forall v \in V(G) \qquad (2.14)$$

Constraints (2.12) allow at most one tree-arc for any edge and none for deleted edges. We force tree nodes to propagate along one outgoing arc by

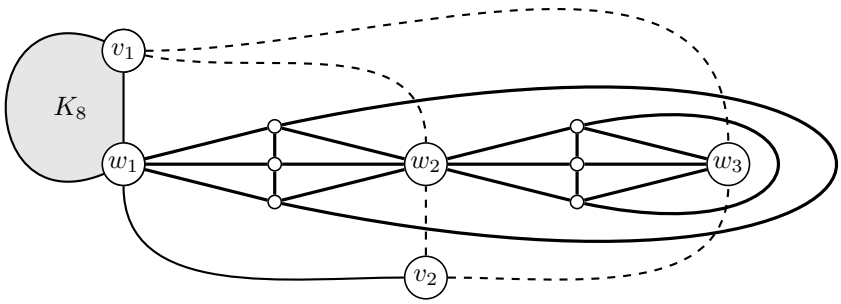


Figure 2.17: Input for proving the strength of the pseudo-tree model. Expensive edges are bold. Removing the dashed edges allows to keep all expensive ones.

constraints (2.13). Finally, constraints (2.14) label degree-1-nodes and nodes where all (but one) neighbor is labeled. Now, we may subtract $\sum_{v \in V(G)} t_v$ nodes (and the same number of edges) from (2.10) to obtain a stronger bound:

Corollary 2.5.10. *The* extended cycle constraint*, given below, is feasible.*

$$(D-1)\big(m - s(E(G))\big) \leq \begin{array}{r} (D+1)(n-2) - 2\sum_{v \in V(G)} t_v \\ + \sum_{d=3}^{D} (D+1-d)c(d) \end{array} \tag{2.15}$$

Alternatively, we may use a less sophisticated approach that does not model propagation but labels only degree-1-nodes. In this case, it suffices to add variables $t_v \in \{0,1\}$, $\forall v \in V(G)$, and constraints (2.14), assuming $\sum_{w \in N(v)} t_{wv} = 0$.

Lemma 2.5.11. *The pseudo-tree extension, i.e., constraints* (2.12)–(2.15) *together with the t-variables, strengthens* CM_3*. This already holds for the approach without propagation.*

Proof. Consider the following input G: start with two K_5's that we each delete an arbitrary edge from and join them by identifying two degree-3 nodes with each other. We call the resulting node w_2. The new graph has exactly 3 nodes $W \coloneqq \{w_1, w_2, w_3\}$ of degree not 4. We denote its edge-set by E_5. We add the nodes v_1, v_2 and connect both to all nodes of W. Let X denote a set of 6 new nodes: we complete a K_8-subgraph on $X \cup \{w_1, v_1\}$ and denote its edges by E_8. Finally, we assign weight w to the edges: fix a (large) constant $M \in \mathbb{N}$; all edges of E_5 have weight M, all other edges have weight 1. See Fig. 2.17 for a schematic depiction of G.

Consider CM_3 on G. We show the fact for $\mathrm{OBJ} = 7$. We observe that all Kuratowski constraints are satisfied by the following s-variable assignment: $s_e = 0$ for all $e \in E_5 \cup \{v_2w_1\}$, $s_{v_1w_2} = s_{v_1w_3} = 17/18$, $s_{v_2w_2} = s_{v_2w_3} = 1$, and $s_e = 1/9$ otherwise. This gives an objective value of 7, independent of M. For this objective, the cycle constraint (2.10) is satisfied if $c(3) \geq 28$ which is obtained by setting $c_\alpha = 1$ for 12 triangles α in E_5 (corresponding to an actual embedding of the two joined K_5-subgraphs) and $c_\alpha = 8/27$ for all $\binom{8}{3}$ triangles α in E_8 without violating constraints (2.9). Note that $\deg(v_2) = 1$ in the solution.

Consider either variant of the pseudo-tree model (2.12)–(2.15) and assume that the objective value would remain feasible. We observe that for any two edges $e, f \in \delta_G(v_2)$, the graph $G[E_5] \boxplus e \boxplus f$ is not planar. For any 6 nodes in $V(G[E_8])$ we obtain a $K_{3,3}$-subdivision. Summing over the respective Kuratowski constraints, it follows that $s(E_8) \geq |E_8|/9$. Note that no edge in $E(G) \setminus (E_5 \cup E_8)$ is part of a triangle. Let $c(F)$ denote the sum of triangles that use edges of $F \subseteq E(G)$. By the extended cycle constraint (2.15) we have $c(3) \geq 28 + 2t_{v_2}$. Since $c(3) \leq c(E_5) + c(E_8) \leq 2/3\big(|E_5| + |E_8| - s(E_8)\big) \leq 2/3\big(18 + 28 - 28/9\big) = 772/27$, we obtain $t_{v_2} \leq 8/27$. Constraints (2.14) imply that $s\big(\delta_G(v_2)\big) \leq t_{v_2} + 1 = 35/27$. Hence, there exists a pair of edges $e, f \in \delta_G(v_2)$ such that $s_e + s_f \leq 35/27 \cdot 2/3 = 70/81$ and $s(E_5) \geq 11/81$ follows. By choosing $M > (7 \cdot 81)/11$ we obtain a contradiction. □

All following constraint classes deal with excluding combinations of cycles and paths that either induce non-planarity, or result in cycle-variables not assignable to any face in the planar subgraph (Lemma 2.5.5). They are independent of but compatible with the pseudo-tree extension.

Cycle-Edge Constraints. Considering constraints (2.9) on feasible (integral) solutions, a cycle cannot be picked if any of its edges is deleted. W.r.t. fractional solutions we can hence additionally require

$$s_e + c_\alpha \leq 1 \qquad \forall \alpha \in \mathcal{C}_{\leq D}, e \in E(\alpha). \tag{2.16}$$

Although there are only $\mathcal{O}(D|\mathcal{C}_{\leq D}|)$ such constraints, preliminary benchmarks showed that adding all of them does not pay off. Instead, we straightforwardly separate them by iterating over the edges of each cycle that has a non-zero variable.

Lemma 2.5.12. *The cycle-edge constraints* (2.16) *strengthen* CM_3.

Proof. Consider the K_7, and pick three nodes $v_1, v_2, v_3 \in V(K_7)$. We add the new nodes $w_1, w_2 \notin V(K_7)$ and new edges $w_1w_2, v_1w_1, v_2w_2, v_3w_1, v_3w_2$

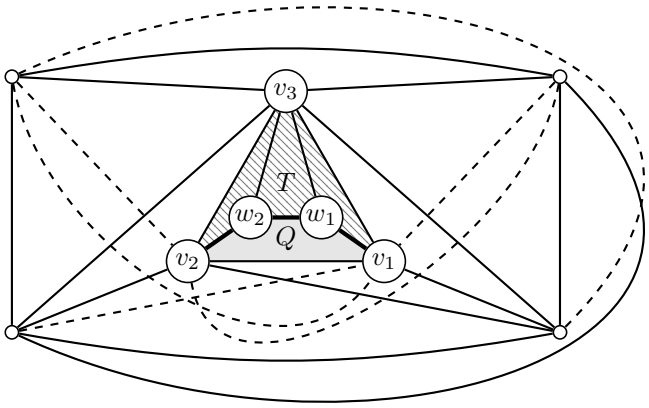


Figure 2.18: Input for proving strength of cycle-edge constraints. The solid edges induce a planar subgraph and show that the same skewness is maintained when inserting w_1, w_2 into the K_7. Quadrangle Q is shaded and the set T of triangles is hatched. All edges incident with exactly one triangle are drawn in bold.

to obtain G, see Fig. 2.18. Note that the edges $I \coloneqq w_1w_2, v_1w_1, v_2w_2$ are incident with exactly one triangle (each with a different one, let T denote the set of the three triangles) and a common quadrangle $Q = v_1w_1w_2v_2$. We observe that $skew(G) = skew(K_7) = 6$.

Consider CM_3 on G. We show the fact for OBJ $= 5$. We choose $s_e = 1/8$ for all edges $e \notin E(Q)$ and set $s_{v_1w_1} = s_{v_2w_2} = 1/2$, $s_{v_1v_2} = s_{w_1w_2} = 5/8$. Without constraints (2.16), this allows us to set $c(3) = 14$ (each edge not in $E(Q)$ being incident with two triangles α, β whose cycle variables are 1 and $3/4$, respectively), satisfying the cycle constraints. It is straightforward to verify the existence of this cycle variable assignment (but manually tedious). Clearly, this bound is no better than the trivial Euler constraint on G.

However, when adding the cycle-edge constraints (2.16), this solution becomes infeasible: for example $s_{v_1w_1} = 1/2$ but $c_{v_1w_1v_3} \geq 3/4$. We will show that the objective is larger than 5 by contradiction: To obtain an objective value of 5, we require $c(3) \geq 14$. Let E_7 denote the edges in the K_7-subgraph of G, $\bar{E}_7 \coloneqq E(G) \setminus E_7$, and $x \coloneqq s(E_7)$. Recall that $s(E(G)) \geq 5$ by Euler and hence $s(\bar{E}_7) \geq 5 - x$. Let $c(F)$ denote the sum of triangles where each triangle is weighted by its number of edges in $F \subseteq E(G)$ divided by 3. Already by summing up constraints (2.9) over all edges of E_7 we have $c(E_7) \leq 2/3\big(|E_7| - x\big) = 14 - 2x/3$ and hence $c(\bar{E}_7) \geq 2x/3$. Note that $c(\bar{E}_7) = \sum_{\alpha \in T} c_\alpha$. For each $\alpha \in T$ with $c_\alpha > 0$, we require a

certain number of edges in $\bar{E}_7$ not to be deleted. More precisely, we may assume the largest two cycle variables of T to be incident with 4 different edges of $\bar{E}_7$ while the smallest one adds just a single edge of $\bar{E}_7$ that is not used in the other two cycles. Assuming the new constraints to be satisfied on I, we obtain $s(\bar{E}_7) \leq |\bar{E}_7| - c(\bar{E}_7)(2/3 \cdot 2 + 1/3)$. Since K_7 contains a Kuratowski subdivision, it follows that $x > 0$ and we have the contradiction $5 - x \leq s(\bar{E}_7) \leq 5 - 10/9x$. □

Two-Cycles-Path Constraints. Given two cycles α and β, we denote their set of inner nodes by $\nu(\alpha, \beta) \coloneqq \{v \in V(\alpha) \cap V(\beta) \mid \delta_\alpha(v) = \delta_\beta(v)\}$. Let $\sigma(\alpha, \beta)$ denote the set of non-empty paths that connect $\nu(\alpha, \beta)$ to $V(\alpha \sqcup \beta)$ without using any edge in $E(\alpha \sqcup \beta)$.

Lemma 2.5.13. *The two-cycles-path constraints, given below, are feasible.*

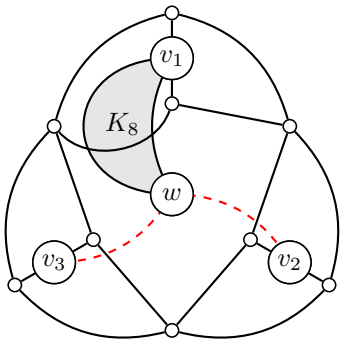
$$s\big(E(p)\big) \geq c_\alpha + c_\beta - 1 \qquad \forall \alpha, \beta \in \mathcal{C}_{\leq D}; p \in \sigma(\alpha, \beta) \tag{2.17}$$

Proof. Assume an embedding π of $\alpha \sqcup \beta$ where each of α, β corresponds to a face in π. By inserting p into π, we either split face α or face β. Hence, even in a supergraph of $\alpha \sqcup \beta \sqcup p$ two such faces cannot exist. Otherwise, if no such π exists, we have $1 \geq c_\alpha + c_\beta$. □

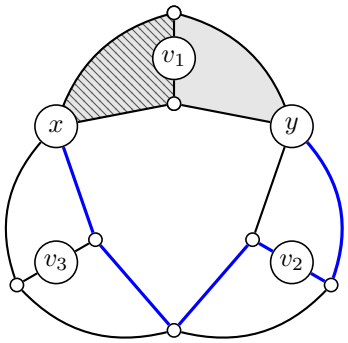
Lemma 2.5.14. *The two-cycles-path constraints* (2.17) *strengthen* CM_4.

Proof. We construct our input G in the following manner, see Fig. 2.19a: Consider the K_3 and replace each of the three edges by a new $K_{2,3}$ such that its end nodes become two nodes of the cardinality-3 node partition of $K_{2,3}$. Let v_1, v_2, v_3 denote the nodes of the cardinality-3 partitions that are not identified with any of $V(K_3)$. We add a new node w and the edges wv_2, wv_3. Finally, we add six new nodes X and a K_8-subgraph on $X \cup \{w, v_1\}$. Let E_8 denote the edge set of this K_8-subgraph. Note that w is not incident with any quadrangle outside of E_8 and all triangles of G are contained in E_8.

Consider CM_4 on G. We show the fact for OBJ = 6. First, note that the graph $G - E_8$ becomes planar when removing any edge incident with w. In fact, all Kuratowski constraints are satisfied by setting $s_e = 1/9$ for all $e \in E_8$ and $s_{wv_2} = s_{wv_3} = 1$. Let $F \coloneqq E(G) \setminus \big(E_8 \cup \delta_G(w)\big)$. To obtain an LP-feasible c-variable assignment, we additionally set $s_e = 4/81$ for all edges $e \in F$, set $c_\alpha = 8/27$ for all triangles α in E_8, and set $c_\alpha = 77/81$ for all 9 quadrangles α in F. It is easy to see that constraints (2.9) and (2.10) are satisfied. Hence, an objective value of 6 is feasible unless the new constraints are added.



(a) The input G with a schematic depiction of the K_8-subgraph. The subgraph $G[V(G) \setminus X]$ without the dashed edges (red) is planar.



(b) Example of a two-cycles-path constraint: if no bold edge (blue) is deleted, the shaded and the (overlapping) hatched quadrangle cannot both occur as part of a face (here, only the hatched one does).

Figure 2.19: Graphs for proving strength of two-cycles-path constraints.

We now show that this is no longer the case when adding the new two-cycles-path constraints (2.17). Assume otherwise, i.e., $s(E(G)) \leq 6$. We denote the sum over all variables of quadrangles in F that are not incident with any v_i ($i \in \{1,2,3\}$) by c_{out}. Similarly, we denote the sum over all variables of quadrangles in F that are incident with some v_i ($i \in \{1,2,3\}$), by c_{in} and the sum over all remaining quadrangle variables (each fully contained in E_8) by $c_8(4)$. The Kuratowski constraints on E_8 imply $s(E_8) \geq |E_8|/9$ and it follows from constraints (2.9) that $3/2 \cdot c(3) + 4/2 \cdot c_8(4) \leq |E_8| - s(E_8)$. Hence, $c(3) + c_8(4) \leq 448/27$. The cycle constraint (2.10) implies $41 \leq 2c(3) + c(4)$.

Let us apply the new constraints: For any $i \in \{1,2,3\}$, pick node v_i and one of its (two) incident quadrangles Q_i that we denote by $\alpha_{\text{in}} \in Q_i$. There is a unique quadrangle α_{out} not incident with v_i but with two v_i's F-neighbors. Let $x \in V(\alpha_{\text{in}}) \cap V(\alpha_{\text{out}})$ denote the unique node with $\delta_{\alpha_{\text{in}}}(x) = \delta_{\alpha_{\text{out}}}(x)$, i.e., the inner node of α_{in} with α_{out}. We denote the single node contained in $V(\alpha_{\text{out}}) \setminus V(\alpha_{\text{in}})$ by y. We observe that any x-y-path p in $G - E_8 - E(\alpha_{\text{out}})$ forms a two-cycle-path constraint with α_{out} and α_{in}, see Fig. 2.19b. Summing over all such constraints (there are exactly 16 paths for two fixed cycles), we obtain $2s(F) \geq 2c_{\text{out}} + c_{\text{in}} - 6$. By constraints (2.9), we have $c_{\text{out}} + c_{\text{in}} \leq 9 - s(F)/2$, an upper bound on $s(F)$. Combining both, we obtain $30 \geq 6c_{\text{out}} + 5c_{\text{in}}$ and hence $c_{\text{out}} + c_{\text{in}} \leq 6$. We finally achieve $41 \leq 2c(3) + c(4) \leq 2\big(448/27 - c_8(4)\big) + c_8(4) + c_{\text{out}} + c_{\text{in}} \leq 2 \cdot 448/27 + 6 < 40$,

a contradiction. □

To identify violated two-cycles-path constraints, we consider each edge e. We collect the set $C(e) = \{\alpha \in \mathcal{C}_{\leq D} \mid e \in E(\alpha) \wedge c_\alpha > 0\}$, and check, for each pair $\alpha, \beta \in C(e)$, whether its sum of LP-values is > 1. If so, we compute the set of inner nodes $\nu := \nu(\alpha, \beta)$ and cache the result for future lookup. If $\nu \neq \varnothing$, we iteratively compute shortest paths following either of two patterns: the *combined* approach searches for shortest paths from ν to $V(\alpha \sqcup \beta) \setminus \nu$, whereas the *separate* one searches for paths from v to $V(\alpha \sqcup \beta) \setminus \{v\}$, separately for each $v \in \nu$. Note that the latter variant will always identify a violated constraint, if one exists, whereas the former ignores paths connecting two inner nodes. After identifying a new path p, an edge in $E(p)$ with maximal LP-value is discarded and the search at v starts anew.

We point out that there is a natural generalization of this constraint class by using k instead of only 2 cycles. If the k cycles fully enclose a common node v (just like two cycles enclosing their inner nodes), any other path from v to the same block is forbidden.

Cycle-Two-Paths Constraints. We say that two paths p_1, p_2 are *conflicting w.r.t. a cycle* α if and only if they each start and end at nodes of $V(\alpha)$ but are otherwise disjoint from α and from one another, and p_2 connects the components of $\alpha[V(\alpha) \setminus V(p_1)]$, cf. Fig. 2.20.

Lemma 2.5.15. *The cycle-two-paths constraints, given below, are feasible.*

$$s\big(E(p_1 \sqcup p_2)\big) \geq c_\alpha \quad \forall \alpha \in \mathcal{C}_{\leq D}, \forall \text{ conflicting paths } p_1, p_2 \text{ w.r.t. } \alpha \quad (2.18)$$

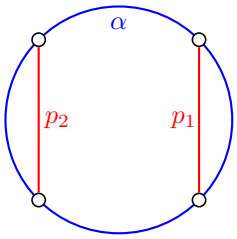
Proof. Given an embedding π of α, we cannot insert both paths p_1, p_2 into the same face of π. Hence, we must split both faces in π. Consequently, no embedding of any supergraph of $\alpha \sqcup p_1 \sqcup p_2$ exists, where there is a face incident with all of α. □

While this constraint class may be stronger than the two-cycles-path constraints, we did not implement it: its separation is complex as we ask for two paths depending on each other.

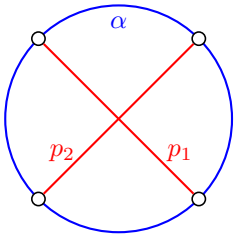
Kuratowski-Cycle Constraints. Starting with a Kuratowski constraint, we can replace parts of its edges by cycles that contain them.

Lemma 2.5.16. *The Kuratowski-cycle constraints, given below, are feasible.*

$$s\big(E(K) \setminus \cup_{\alpha \in C} E(\alpha)\big) \geq 1 - |C| + \textstyle\sum_{\alpha \in C} c_\alpha \quad \forall K \in \mathcal{K}(G), C \subseteq \mathcal{C}_{\leq D} \quad (2.19)$$



(a) Paths of a cycle that are not conflicting.



(b) Conflicting paths of a cycle.

Figure 2.20: Different configurations of disjoint paths $p1, p2$ (red) w.r.t. cycle α (blue). Potentially present subdivision nodes along the cycle and paths are omitted in the pictures. In case (b), cycle α may not appear as part of a face unless at least one path is partially removed.

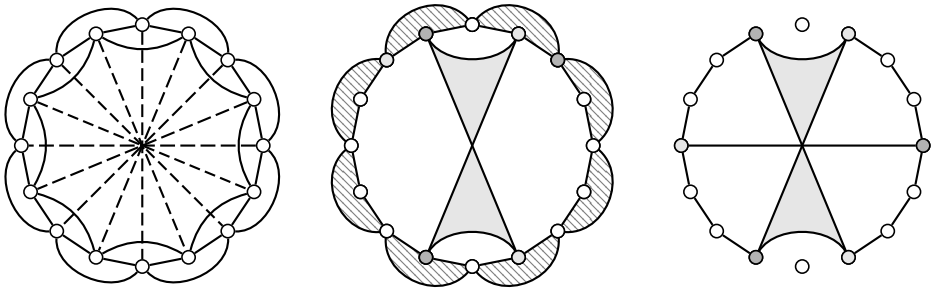
Proof. If $C = \varnothing$, we simply obtain a Kuratowski constraint. Assuming integrality and $C \neq \varnothing$, the right-hand side is 1 if all cycles in C are picked and ≤ 0 otherwise. In the former case, the edges of C, together with the remaining edges of K that are not contained in C contain a Kuratowski subdivision, and we need to remove an edge. □

Lemma 2.5.17. *The Kuratowski-cycle constraints* (2.19) *strengthen* CM_4.

Proof. Let G denote the circulant graph on 16 nodes with jumps $1, 2$, and 8 (see Fig. 2.21a).

Consider CM_4 on G. We show the fact for OBJ $= 14/3$. Assume the nodes of G to be labeled as $v_0, v_1, ..., v_{15}$, corresponding to jumps 1. Let $E_i \subseteq E(G)$ denote the edges corresponding to jumps i. We observe that every Kuratowski subdivision needs at least two edges of E_8 as $(V(G), E_1 \cup E_2 \cup \{e\})$ is planar for all $e \in E_8$ (see Fig. 2.21a). It follows that every Kuratowski constraint is satisfied by $s_e = 1/2$ for all $e \in E_8$. Without the new constraints, an LP-value of $14/3$ is obtainable by $s_e = 1/2$ for all $e \in E_8 \cup \{s_{v_1v_3}\}$, $s_{v_5v_7} = 1/6$, and $s_e = 0$ otherwise; and by picking all 16 triangles and four disjoint quadrangles R, consisting of two edges of $E_2 \setminus \{v_1v_3, v_5v_7\}$ and E_8 each, with cycle variable value 1.

The solution is cut-off by the new Kuratowski-cycle constraints (2.19): intuitively, we may pick a quadrangle of R instead of two edges of E_8 to obtain a violated Kuratowski-cycle constraint although the sum of any two such edges being 1 prevents any traditional Kuratowski constraints to be violated.



(a) Input: circulant graph with jumps 1 (solid straight), 2 (solid bent), and 8 (dashed): adding any single edge of E_8 to $G[E_1 \cup E_2]$ maintains in a planar graph.

(b) This subgraph contains a $K_{3,3}$-subdivision and consists of eight triangles (hatched) and a 2-8-quadrangle (shaded).

(c) $K_{3,3}$-subdivision that consists of 12 edges of E_1, one edge of E_8 and a disjoint 2-8-quadrangle (shaded).

Figure 2.21: Graphs for showing the strength of Kuratowski-cycle constraints.

We now show that $s(E(G)) > 14/3$ when adding a certain set of Kuratowski-cycle constraints. An i-j-quadrangle is a quadrangle that uses edges of E_i and E_j. Let q_{ij} denote the sum over c-variables corresponding to i-j-quadrangles. Observe that $c(4) = q_{12} + q_{18} + q_{28}$, since there are no other quadrangles in G. The cycle constraint (2.10) implies that $3s(E(G)) + 2c(3) + q_{12} + q_{18} + q_{28} \geq 50$. By summing over all constraints (2.9) that correspond to edges of E_1, we obtain $c(3) + q_{12} + q_{18} + s(E_1) \leq 16$. Similarly, using E_8 instead of E_1, we have $q_{18} + q_{28} + s(E_8) \leq 8$.

We describe three types of Kuratowski-cycle constraints (2.19) in G. Let W denote the nodes of G that have an odd index. We observe that $G[W]$ is a Möbius ladder: it consists of exactly four 2-8-quadrangles and is non-planar. The same construction works for even node indices. We obtain $q \leq 6$ by summing over both constraints. Consider a single 2-8-quadrangle α: the indices of all nodes in α have identical parity p. The graph that consists of α and the 8 triangles that use edges of E_2 incident with nodes of parity not p, is non-planar, see Fig. 2.21b. Summing over all corresponding Kuratowski-cycle constraints, we obtain $64 \geq 4c(3) + q_{28}$. Once again, consider a single 2-8-quadrangle α: there are exactly two nodes $u, v \notin V(\alpha)$ each incident with two nodes of $V(\alpha)$. Pick any $f \in E_8 \cap E(\alpha)$ such that $f \cap \{u, v\} = \varnothing$. The graph $\alpha \sqcup G[f \cup \{e \in E_1 \mid e \cap \{u, v\} = \varnothing\}]$ is non-planar, see Fig. 2.21c. It consists of a 2-8-quadrangle, another edge of E_8, and 12 edges of E_1. Summing over all corresponding Kuratowski-cycle constraints, we have $6s(E_1) + s(E_8) \geq q_{28}$.

By the above, we obtain a system of linear inequalities on six non-negative variables:

$$\begin{aligned} s(E_1) + s(E_2) + s(E_8) &\leq 14/3 \\ 3s(E_1) + 3s(E_2) + 3s(E_8) + 2c(3) + q_{12} + q_{18} + q_{28} &\geq 50 \\ c(3) + q_{12} + q_{18} + s(E_1) &\leq 16 \\ q_{18} + q_{28} + s(E_8) &\leq 8 \\ q &\leq 6 \\ 4c(3) + q_{28} &\leq 64 \\ 6s(E_1) + s(E_8) &\geq q_{28} \end{aligned}$$

It is straightforward (although manually tedious) to check that this system has no solution. □

For separation, we identify a Kuratowski subdivision K as for (2.1). We collect the set S of cycles with LP-value > 0 incident with K. For each cycle in S, we compute its *gain*, i.e., the increase in violation (or decrease in slack) when adding that cycle to C. As long as there are cycles with positive gain, we continue adding a cycle of S with maximal gain to C.

Cycle-Clique Constraints. Two cyclic orders $\pi, \bar{\pi}$ on a set X are *conflicting* if and only if $\pi \neq \bar{\pi}$ and $\pi \neq \text{reverse}(\bar{\pi})$. The restriction of π to $Y \subseteq X$ is denoted by π^Y. A cycle α induces a unique cyclic order on its nodes $V(\alpha)$ (up to reversal). Given two cycles α, β, let π_α, π_β be corresponding cyclic orders, and let $W \coloneqq V(\alpha) \cap V(\beta)$ be the common nodes. We say that α and β are *conflicting* if and only if π_α^W and π_β^W are conflicting.

Lemma 2.5.18. *The cycle-clique constraints, given below, are feasible.*

$$\sum_{\alpha \in C} c_\alpha \leq 1 \quad \forall C \subseteq \mathcal{C}_{\leq_D} \text{ s.t. all cycles in } C \text{ are pairwise conflicting} \tag{2.20}$$

Proof. Consider any pair of conflicting cycles $\alpha, \beta \in C$ with π_α, π_β, and W defined as above. Since cyclic orders on three elements are unique up to reversal, we have $|W| \geq 4$. By transitivity there exists a set of exactly four common nodes $X \subseteq W$, such that π_α^X and π_β^X are conflicting. The graph on X where we add an edge vw if and only if v is adjacent to w in π_α^X or π_β^X is the K_4. Since the K_4 is not outerplanar, there can neither be a face in K_4 traversing all of X nor such a face in $\alpha \sqcup \beta$. □

We create the conflict graph H_C that contains a node for every cycle with LP-value > 0, cache the conflict information for each pair of cycles, and add constraints for maximal cliques in H_C. In a less sophisticated variant, we only add constraints for "cliques" of size two.

2.5.3 Experiments

All algorithms are implemented in `C++`, compiled with GCC 6.3.0, and use the OGDF (snapshot 2017-07-23) [Chi+13]. We use SCIP 4.0.1 for solving ILPs with CPLEX 12.7.1 as the underlying LP solver [Gle+18]. Each MPS-computation uses a single physical core of a Xeon Gold 6134 CPU (3.2 GHz) with a memory speed of 2666 MHz. We employ a time limit of 20 minutes and a memory limit of 8 GB per computation. Our instances and results, giving running time and skewness (if solved), are available for download at `http://tcs.uos.de/research/mps`.

Instances and Algorithms. Analogously to our previous studies, and as announced in Section 1.7.1, we consider the three real-world sets North, Rome, and a subset of SteinLib, as well as the artificial set Regular. For tuning of ε (e.g., heap size in separation) we rely on the values identified in [Hed17]. We use the notation specified in Table 2.8 to denote algorithmic choices. Note that instead of providing D explicitly, we specify a minimum number r' of cycle variables to be generated. We increment D while there are less than r' many.

Results. Table 2.9 shows the success rates (percentage of instances solved to proven optimality) and average running time per instance of our algorithmic variants. For non-solved instances we assume the maximum running time of 20 minutes—average running times are thus comparable only for algorithms that achieve roughly equal success rates. We group the variants by the number of used extensions and highlight variants that dominate their group in bold. The latter informs our choice of which variants to consider in the next group.

The separation of generalized Euler constraints is clearly beneficial only on the North graphs, but even there its improvements are marginal when compared to the cycle-based approach. The latter works very well in practice, for all instance sets. In particular (cf. Fig. 2.22), on Rome it allows us for the first time to compute the skewness of *all* instances. Using variant *c10 t0 i w0*, we are able to solve all but `grafo10958.98.lgr` within the 20 minute time frame; this last instance required 103 minutes. North still

Table 2.8: Notation to encode algorithmic variants.

ε	Do not use any extensions but the basic Kuratowski algorithm [Mut94].
e	Separate generalized Euler constraints (2.7).
c{r}	Add cycle constraints (2.9), (2.10), and variables with the minimal value for D such that there are variables for at least $100r$ cycles.
t{0|1}	Use the pseudo-tree extension (2.12)–(2.15) with (=t1) or without (=t0) propagation.
i	Enforce integrality of variables for cycles and pseudo-trees.
s	Separate cycle-edge constraints (2.16).
w{0|1}	Separate two-cycles-path constraints (2.17) using *combined* (=w0) or *separate* (=w1) approach. Also enables separation on cycle-clique constraints (2.20) for 2-cliques.
k	Separate Kuratowski-cycle constraints (2.19).
q	Separate cycle-clique constraints (2.20).

Table 2.9: Overview of performance for algorithmic variants: success rate in percent of solved instances and average running time in seconds. Fastest and most successful variants are highlighted in bold within each block.

	Rome		North		Regular		SteinLib	
	succ.	time	succ.	time	succ.	time	succ.	time
ε	85.71	198.42	73.76	325.31	34.74	800.38	9.52	1 085.94
e	85.56	199.41	77.78	273.29	35.00	803.91	9.52	1 085.93
c5	98.91	21.60	84.40	201.42	53.95	567.52	31.43	859.40
c10	**99.14**	**18.10**	**84.63**	**195.35**	54.47	562.81	**32.38**	**853.57**
c20	**99.14**	19.58	83.92	197.86	**55.00**	573.82	31.43	861.47
c10 i	99.89	5.52	88.89	156.99	56.58	538.81	31.43	841.88
c10 s	99.79	6.66	88.42	165.54	**58.68**	**515.13**	35.24	821.00
c10 t0	**99.95**	**3.36**	92.43	112.82	57.37	535.46	**37.14**	789.26
c10 t1	99.92	3.74	**93.14**	**111.94**	56.32	539.04	**37.14**	**785.89**
c10 w0	99.79	7.07	87.23	165.36	55.53	549.94	31.43	837.52
c10 w1	99.82	6.63	86.52	179.48	55.00	553.38	31.43	833.81
c10 k	99.77	7.26	86.52	178.34	55.26	552.83	33.33	828.81
c10 q	99.73	7.51	85.82	185.84	55.53	550.55	31.43	841.77
c10 t0 i	99.95	3.22	93.14	109.61	57.37	529.93	38.10	782.72
c10 t0 s	**99.98**	3.08	**93.62**	**95.46**	**58.95**	**509.01**	**39.05**	**760.70**
c10 t0 w0	**99.98**	**2.75**	92.43	112.77	57.37	537.61	36.19	808.58
c10 t0 w1	**99.98**	2.92	92.20	109.37	57.11	537.76	37.14	780.83
c10 t0 k	99.92	3.57	92.67	104.55	56.84	535.97	38.10	785.46
c10 t0 q	99.95	3.58	92.67	109.71	57.37	538.19	37.14	789.05
c10 t1 i	99.96	3.32	92.91	106.39	57.11	533.51	37.14	788.52
c10 t1 s	**99.98**	**2.75**	92.43	112.77	58.68	537.61	37.14	808.58
c10 t1 w0	**99.98**	3.04	92.20	114.28	56.84	537.97	38.10	786.93
c10 t1 w1	**99.98**	3.19	91.96	113.03	56.84	540.39	37.14	783.09
c10 t1 k	99.92	3.65	92.20	112.61	56.84	538.91	37.14	784.30
c10 t1 q	99.92	3.86	93.14	113.89	56.05	540.23	37.14	788.07
c10 t0 i s	99.94	3.27	92.91	103.17	**58.95**	506.63	**40.00**	761.47
c10 t0 s w0	**99.98**	2.43	**93.85**	**91.66**	58.68	508.28	39.05	763.08
c10 t0 s w1	**99.98**	**2.29**	92.91	101.17	58.68	507.99	39.05	756.31
c10 t0 s k	99.96	3.03	93.38	98.06	58.68	504.54	39.05	765.08
c10 t0 s q	99.93	3.22	93.62	95.59	58.42	510.38	38.10	763.64
c10 t0 i w0	**99.99**	2.89	92.67	105.16	57.11	529.95	38.10	798.26
c10 t0 i s w0	99.96	2.72	**94.33**	**93.99**	**59.47**	**502.30**	**39.05**	**754.46**

Table 2.10: Relative improvement over ε for selected algorithmic variants. We give the the success rate over the instances unsolved by ε in percent, and the average running time ratio over the commonly solved instances.

	Rome		North		Regular		SteinLib	
	new	speed	new	speed	new	speed	new	speed
c10	93.98	**66.80**	42.34	21.45	30.24	13.96	25.26	**11.79**
c10 t0 i w0	**99.92**	60.85	72.07	28.59	34.27	12.68	31.58	6.79
c10 t0 i s w0	99.75	59.58	**78.38**	**34.03**	**37.90**	**23.21**	**32.63**	5.42

Table 2.11: Average number of cycle variables and average values for maximum cycle length D.

	Rome		North		Regular		SteinLib	
	D	# var	D	# var	D	# var	D	# var
c5	9.51	627	7.34	689	5.43	2 075	5.80	881
c10	10.51	1 168	8.01	1 213	5.73	2 816	6.64	3 658
c20	11.51	2 175	8.55	2 048	6.47	7 774	7.09	4 785

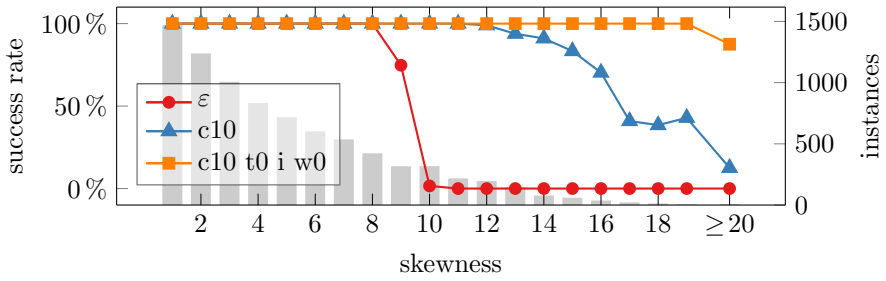
contains instances too hard to solve exactly (even when increasing running time to a few days and memory to 32GB). Nonetheless, we now solve 3/4 of the previously unsolved NORTH graphs within our strict limits. The second group of variants in Table 2.9 demonstrates that all of our extensions of the cycle model, in particular the pseudo-tree approach, improve upon success rate and running time on all instance sets when applied to *c10*. As shown in the lower sections of the table, this does not always apply when comparing models that simultaneously use multiple extensions.

Table 2.10 details the relative improvement for each of the three most promising algorithm configurations over the state-of-the-art ε-model. We provide the success rate for the instances not solved by ε and give the average relative speed-up (i.e., the running time of ε divided by that of variant X) over the instances solved by both ε and X. This common set is exactly those solved by ε, except for a single ε-solved NORTH-instance not solved by *c10*. On ROME, the pure cycle model *c10* without any further extensions achieves the best speed-up; for the seemingly harder other instance sets, more sophisticated variants are worthwhile. Fig. 2.22 underlines that the success rate of the algorithms is strongly correlated to the instance's skewness.

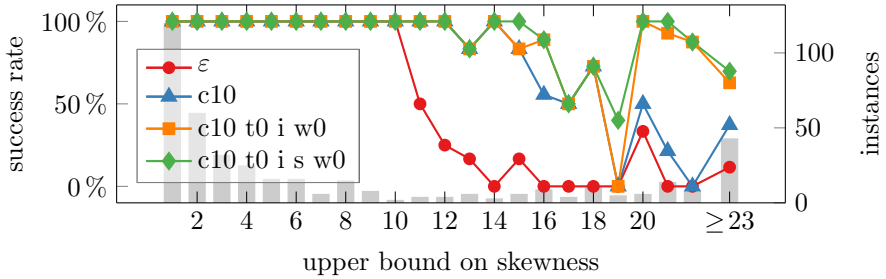
Table 2.11 lists the average number of generated cycle variables and the respective average values for D. We mention that instances with high D values typically generate *few* cycle variables, close to the lower bound. However, there is a large deviation in the number of generated cycle variables in any fixed instance set: some graphs contain less than the requested number of cycles whereas others already contain roughly 10 000 triangles.

Conclusion. For over two decades, the strongest ILP model for MPS has not been improved. In this section we presented novel variables and constraints, based on cycles, to extend this model to finally obtain both a theoretically stronger model, as well as a more efficient algorithm in practice. We proved that there is a hierarchy of ever stronger LP-relaxations, induced by the maximal considered cycle length, and a rich set of further strengthening cycle-based constraint classes. For the first time, we are able to compute the skewness of the entire set ROME, solve 94% of the NORTH graphs (compared to 74% by the ε-model), and solve 40% instead of only 10% of our STEINLIB instances. Our extensions also help for the notoriously hard expander graphs (REGULAR).

Several of our proofs show the new constraint class's strength w.r.t. a low-D cycle model. We conjecture that most classes remain strengthening for high D, but to prove this, one has to find and argue infinite families of



(a) Solved ROME graphs by skewness. Algorithm *c10 t0 i w0* solves all but a single graph that has skewness 22 within 20 minutes.



(b) Solved NORTH graphs by best upper bound on skewness.

Figure 2.22: Detailed success rates for selected algorithmic variants.

graphs with the LP-properties of our currently hand-crafted proof graphs. Furthermore, it is natural to ask if and which of the new constraint classes form facets in the (lifted) planar subgraph polytope.

A problem inherent to our approach arises on inputs of non-homogeneous density: G may have too dense subgraphs to raise D sufficiently, even when every planar subgraph of G contains large regions consisting of high-degree faces. Is there a practical way to generalize the cycle-based approach using an independent maximum cycle length *for each edge*?

Chapter 3

Genus

Cycle (*music*). The set of pitch classes resulting from repeatedly applying the same interval class to the starting pitch class.

In this chapter, we will derive stronger ILP models for the graph genus problem that allow us to compute the genera of general graphs. Similarly to skewness, cf. Section 2.5, we are going to consider small cycles and their natural generalization, i.e., closed walks, in the input.

The first ILP and PBS models were published just recently [Bey+16], and we will use the respective ILP model as the basis for our algorithms as well as the baseline for our practical evaluation.

Just like the known model, ours will simulate the face tracing algorithm. As such, both models share a common foundation that we borrow from [Bey+16]. For the sake of completeness, we repeat its definition below: We use variables x_i that are 1 if and only if face i exists, and variables x_i^a that are 1 if and only if arc a participates in the boundary of face i. Let $\bar{f}$ be an upper bound on the number of faces. We use the shorthands $x(I, A) \coloneqq \sum_{i \in I, a \in A} x_i^a$ and $x(A) \coloneqq x([\bar{f}], A)$; thereby, we may omit curly braces when providing sets of cardinality one. Consider the following (by

itself insufficient!) model (3.1a–3.1e) that we call $\mathrm{ILP}_{\mathrm{Base}}$.

$$\begin{aligned}
\max \quad & \textstyle\sum_{i=1}^{\bar{f}} x_i && && (3.1a)\\
\text{s.t.} \quad & 3x_i \leq x(i, A) && \forall i \in [\bar{f}] && (3.1b)\\
& x(a) = 1 && \forall a \in A && (3.1c)\\
& x(i, \delta_-(v)) = x(i, \delta_+(v)) && \forall i \in [\bar{f}],\ v \in V && (3.1d)\\
& x_i, x_i^a \in \{0, 1\} && \forall i \in [\bar{f}],\ a \in A && (3.1e)
\end{aligned}$$

Following [Bey+16], $\mathrm{ILP}_{\mathrm{Base}}$ ensures that the faces form a partition of the arc set such that each cell consists of at least three arcs and is a collection of closed walks.

It remains to ensure that the faces are consistent with some rotation scheme of the edges around the nodes. In [Bey+16], this is achieved via *predecessor* variables, and we denote the model by $\mathrm{ILP}_{\mathrm{Pre}}$ in the following. It uses $\mathrm{ILP}_{\mathrm{Base}}$ and additionally (3.2a–3.2d). The idea is to establish a cyclic order of the incident edges of each node by a cut-based sub-ILP known from the traveling salesman problem. The cyclic rotation around each node v is encoded by variables $p^v_{u,w}$ that are 1 if and only if arc uv directly precedes arc vw when tracing the respective face.

$$\begin{aligned}
x_i^{vw} &\geq x_i^{uv} + p^v_{u,w} - 1\\
x_i^{uv} &\geq x_i^{vw} + p^v_{u,w} - 1
\end{aligned} \qquad \forall i \in [\bar{f}],\ v \in V,\ u, w \in \mathrm{N}(v) : u \neq w \qquad (3.2a)$$

$$\begin{aligned}
\textstyle\sum_{\substack{u \in \mathrm{N}(v):\\ u \neq w}} p^v_{w,u} &= 1\\
\textstyle\sum_{\substack{u \in \mathrm{N}(v):\\ u \neq w}} p^v_{u,w} &= 1
\end{aligned} \qquad \forall v \in V,\ w \in \mathrm{N}(v) \qquad (3.2b)$$

$$\textstyle\sum_{u \in W, w \in \mathrm{N}(v) \setminus W} p^v_{u,w} \geq 1 \qquad \forall v \in V,\ W \subsetneq \mathrm{N}(v) : \varnothing \neq W \qquad (3.2c)$$

$$p^v_{u,w} \in \{0, 1\} \qquad \forall v \in V,\ u, w \in \mathrm{N}(v) : u \neq w \qquad (3.2d)$$

Constraints (3.2a) ensure that if an arc uv is contained in a given face i and uv is succeeded by vw, then vw must be contained in the same face and vice versa. Each arc requires exactly one direct predecessor and successor as modeled by constraints (3.2b). Finally, the rotation around v induced by $p^v_{\cdot\cdot}$ must consist of exactly one cycle. This is modeled by the (exponentially many) subtour elimination (or *cut*) constraints (3.2c), as known from the standard ILP for the traveling salesman problem.

3.1 Realizability Model

In contrast to the explicit modeling of an embedding in $\mathrm{ILP}_{\mathrm{Pre}}$, we establish the existence of an embedding *implicitly*. Our *realizability constraints* (see (3.3) later) require only the variables of $\mathrm{ILP}_{\mathrm{Base}}$. We first need some auxiliary concepts. A graph $G = (V, A)$ is *loopy* if it is directed, connected, each node has at least one incoming arc, and for each arc $uv \in A$ it holds that $\mathcal{K}(uv) \coloneqq G[\{s : sv \in A\} \cup \{t : ut \in A\}]$ is a complete bipartite graph $K_{k,k}$ for some $k \in \mathbb{N}$ such that each arc is directed from the cell of u to that of v w.r.t. the bipartition.

Lemma 3.1.1. *Loopy graphs are Hamiltonian, i.e., they contain a cycle traversing all nodes.*

Proof. We first show that any loopy graph allows a cycle cover of pairwise node-disjoint cycles. Assume it does not, consider a collection $\mathcal{C}$ of pairwise node-disjoint cycles covering as many nodes as possible, and let v be an uncovered node. By loopiness, there exists a bipartition that induces two cells, an arc uv, and $\mathcal{K}(uv)$ has a node w (possibly $u = w$) in the cell of u, such that w is not contained in any cycle of $\mathcal{C}$: for any ℓ nodes of one cell in a cycle $c \in \mathcal{C}$, c also contains ℓ nodes of the other cell. As there are only finitely many nodes, we find a new cycle by iterating our argument, i.e., traversing the cycle's arcs in reverse order, thus increasing our cycle cover; a contradiction.

Now, let $\mathcal{C}$ be a node-disjoint cycle cover. For a cycle $c \in \mathcal{C}$, an arc a connecting $V(c)$ with $V \setminus V(c)$ exists by connectivity. Hence, $\mathcal{K}(a)$ contains an arc uv of c and another arc wx of a different cycle $c' \in C$. We join c with c' to a single cycle by replacing uv, wx with ux, wv. Iterating this yields the claim. □

Theorem 3.1.2. *A graph G allows an embedding Π with at least ξ faces if and only if there exists a partition P of $A(G)$, such that*

(a) P consists of at least ξ cells;

(b) every cell of P is a set of pairwise node-disjoint closed walks; and

(c) for all subsets $X \subseteq P$, nodes $v \in V(G)$, and non-empty subsets $W \subsetneq \mathrm{N}(v)$, we have $\{wv, vw : w \in W\} \neq \bigcup_{x \in X} \{a \in x : v \text{ incident to } a\}$.

Before giving the formal proof, let us provide some intuition on property (c): It models that the rotation around each node v is consistent. While in $\mathrm{ILP}_{\mathrm{Pre}}$ constraints (3.2a–3.2d) model the rotation explicitly, property (c)

ensures the *existence* of a feasible rotation by preventing subcycles. In the rotation around v, any two subsequent arcs must share an edge or a face. Hence, there cannot exist a *proper* subset W of v's neighbors, such that *exactly* the arcs between v and W belong to a subset X of faces. As shown below this is also sufficient.

Proof (of Theorem 3.1.2). Let us establish both directions of the equivalence:

($\Longrightarrow$) Let Π denote the considered embedding with at least ξ faces. We obtain P by creating a cell for each face f of Π: it contains exactly the arcs traversed by f. This satisfies (a) and (b). Assume that (c) is not satisfied, i.e., there exist X, v, W (following the above selection rules) such that $\{wv, vw : w \in W\} = \bigcup_{x \in X} \{a \in x : v \text{ incident to } a\}$. Since W is a proper subset of $\mathrm{N}(v)$, X cannot span all faces incident with v. We choose any face contained (not contained) in X and denote it by f (resp. g). Since Π is an embedding, there exists a sub-sequence of edges incident with v that corresponds to a dual path from f to g. But according to (c), all edges incident with X join two faces in X.

($\Longleftarrow$) Assume, on the other hand, that a partition P exists such that (a)–(c) are satisfied. We find an embedding Π by forming a face from each component of each cell of P. We establish feasible rotations around each node v in the following way: Let D_v be a directed graph with nodes $\mathrm{N}(v)$ such that $uw \in A(D_v)$ if and only if uv and vw are in the same cell of P (i.e., the arcs could be traversed in that order when tracing the face corresponding to v's component of the cell). A feasible rotation around v corresponds to a Hamiltonian cycle in D_v. We show that D_v is loopy, and hence Hamiltonian by Lemma 3.1.1: By construction of D_v, $\mathcal{K}(a)$ is a $K_{k,k}$ for some $k \in \mathbb{N}$, for each $a \in A(D_v)$. By property (b), all nodes of D_v have at least one incoming arc. For disconnected D_v, let W denote the nodes of a single component of D_v, and X the cells that induce $A(D_v[W])$. Then X, v, W contradict property (c).

□

The above theorem shows that it suffices to optimize over all partitions of arcs into faces. Given a feasible partition (w.r.t. Theorem 3.1.2), a corresponding embedding is easily determined in polynomial time following our proof. We can now establish our new model $\mathrm{ILP}_{\mathrm{Real}}$, which extends $\mathrm{ILP}_{\mathrm{Base}}$ with constraints (3.3). While the former already establishes properties (a) and (b), the latter models property (c): the connectivity of the "local dual

graph" D_v around each primal node. Here, index set I corresponds to set X from Theorem 3.1.2.

$$\begin{gathered} x\big(I, v \times_A (\mathrm{N}(v) \setminus W)\big) \geq 1 + x(I, v \times_A W) - 2|W| \\ \forall v \in V, I \subseteq [\bar{f}], W \subsetneq \mathrm{N}(v) : W \neq \varnothing \end{gathered} \tag{3.3}$$

Separation. Clearly, it is impractical to add all exponentially many constraints (3.3) when solving $\mathrm{ILP}_{\mathrm{Real}}$. We use a heuristic separation routine to identify a relevant subset of these constraints. For each LP-feasible solution encountered during the solving process, we proceed as follows: For each node v, we check if all variables x_a^i of its incident arcs a are integral. If this holds but the corresponding D_v is disconnected, we found a new violation of (3.3).

3.2 Small Faces

The following approach is inspired by the cycle model for the maximum planar subgraph problem [CW18] that we presented in Section 2.5.1. There, a mapping between small faces and short cycles was used to (only) strengthen the LP-relaxation of another, by itself sufficient, model. Thus, it was possible to mostly disregard longer cycles. In the genus setting we have to be more careful: On the one hand, we need to consider a far wider range of drawings as we embed on surfaces of higher genera. On the other hand, we have to directly adapt the core model itself; to continue to have a sufficient model, we need to *precisely* encode *all*, even very large, faces. We will model "short" faces by new binary y-variables, one for each specific feasible set of arcs. We continue to use the x-variables for *generic*, i.e., "large" faces. On sparse graphs, this yields a reduction of x-variables, as we may drastically decrease the upper bound on the number of generic faces. Both models, $\mathrm{ILP}_{\mathrm{Pre}}$ and $\mathrm{ILP}_{\mathrm{Real}}$, can be extended in this way.

Let $\mathcal{C}_\sigma$ denote the maximal set of closed walks such that each walk's length satisfies property σ. Note that—in contrast to the cycle model for skewness—we consider all closed walks here, not just cycles. Expanding on $\mathrm{ILP}_{\mathrm{Base}}$, we parameterize our new model $\mathrm{ILP}_{\mathrm{Base}}^D$ by some $D \geq 2$ and obtain (3.4a–3.4g) below. We introduce a new decision variable y_c for each $c \in \mathcal{C}_{\leq D}$ that is 1 if and only if the respective closed walk c is the boundary of a face in the embedding. Let $y(a) \coloneqq \sum_{c \in \mathcal{C}_{\leq D} : a \in c} y_c$ and $\bar{f}_{>d}$ denote any upper

bound on the number of faces with length greater than d.

$$\begin{aligned}
\max \quad & \textstyle\sum_{i=1}^{\bar{f}_{>D}} x_i + \sum_{c \in \mathcal{C}_{\leq D}} y_c && && \text{(3.4a)}\\
\text{s.t.} \quad & (D+1)x_i \leq x(i, A) && \forall i \in [\bar{f}_{>D}] && \text{(3.4b)}\\
& x(a) + y(a) = 1 && \forall a \in A && \text{(3.4c)}\\
& x(i, \delta_-(v)) = x(i, \delta_+(v)) && \forall i \in [\bar{f}_{>D}], v \in V && \text{(3.4d)}\\
& \textstyle\sum_{i=1}^{\bar{f}_{>D}} x_i + \sum_{c \in \mathcal{C}_{\leq D} : |c| > d} y_c \leq \bar{f}_{>d} && \forall d \in \{2, ..., D\} && \text{(3.4e)}\\
& x_i \in \{0,1\}, x_i^a \in \{0,1\} && \forall i \in [\bar{f}_{>D}], a \in A && \text{(3.4f)}\\
& y_c \in \{0,1\} && \forall c \in \mathcal{C}_{\leq D} && \text{(3.4g)}
\end{aligned}$$

Each arc is contained either in one of the generic faces that each form a set of closed walks (as for $\mathrm{ILP}_{\mathrm{Base}}$), or in a closed walk c with dedicated variable y_c, see constraints (3.4c). Generic faces are large, as required by constraints (3.4b). Constraints (3.4d) are essentially (3.1d). Albeit not required for integral solutions, constraints (3.4e) enforce the previously implicit upper bound on the total number of faces and bound the number of gradually smaller faces.

Predecessor Model. To obtain $\mathrm{ILP}^D_{\mathrm{Pre}}$, we add equations (3.2a–3.2d) to $\mathrm{ILP}^D_{\mathrm{Base}}$, i.e., the same set as for the transition from $\mathrm{ILP}_{\mathrm{Base}}$ to $\mathrm{ILP}_{\mathrm{Pre}}$. Additionally, we require

$$\textstyle\sum_{c \in \mathcal{C}_{\leq D} : uv, vw \in c} y_c \geq p^v_{u,w} - x([\bar{f}], uv) \qquad \forall v \in V,\ u, w \in \mathrm{N}(v). \tag{3.5}$$

Similar to (3.2a), this ensures that if an arc uv is contained in a face modeled by a y-variable, the succeeding arc vw has to be contained in the same face.

Realizability Model. We obtain $\mathrm{ILP}^D_{\mathrm{Real}}$ by starting with $\mathrm{ILP}^D_{\mathrm{Base}}$ and adding the following constraints to realize property (c) of Theorem 3.1.2.

$$\begin{aligned}
x(I, \overline{A^v_W}) \geq 1 + x(I, A^v_W) + \textstyle\sum_{c \in \mathcal{C}_{\leq D} : c \cap \overline{A^v_W} = \varnothing} |c \cap A^v_W| y_c - 2|W| \\
\forall v \in V, I \subseteq [\bar{f}], W \subsetneq \mathrm{N}(v) : W \neq \varnothing, \\
\text{where } A^v_W \coloneqq v \times_A W,\ \overline{A^v_W} \coloneqq v \times_A (\mathrm{N}(v) \setminus W)
\end{aligned} \tag{3.6}$$

They ensure there is no subset W of arcs at a common node v that is fully assigned to a set (consisting of I and a subset of $\mathcal{C}_{\leq D}$) of face variables that do not have an arc outside of W.

***D*-Hierarchy: Strength of LP-Relaxations.** Clearly, $\mathrm{ILP}_{\mathrm{Base}} = \mathrm{ILP}^2_{\mathrm{Base}}$. The value of $\bar{f}$ has a strong influence on the dual bounds obtained by LP-relaxations. Hence, we describe how to determine $\bar{f}$ and $\bar{f}_{>D}$ on general graphs. Let $n \coloneqq |V(G)|$ and $m \coloneqq |E(G)|$. Let $f_{\mathrm{UB}}(a,b) \coloneqq \min\{a, b - \mathbb{1}_{a-b=1 \mod 2}\}$, $\bar{f} \coloneqq f_{\mathrm{UB}}(m-n, \lfloor 2m/3 \rfloor)$, $\bar{f}_{>2} \coloneqq \bar{f}$, and finally $\bar{f}_{>d} \coloneqq \min\{\bar{f}_{>d-1}, \lfloor 2m/(d+1) \rfloor\}$ for $d > 2$. The validity of these bounds follows directly from Euler's formula (assuming that the graph G is non-planar). We are not aware of any better, general, dual bounds. In the following comparison of LP-relaxations we always assume the above bounds.

Lemma 3.2.1. *For every graph G, the LP-relaxation of* $\mathrm{ILP}_{\mathrm{Base}}$ *has objective value $\bar{f}$.*

Proof. The domains (3.1e) establish $\bar{f}$ as an upper bound. Set $\tilde{x}^i_a = 1/\bar{f}$ and $\tilde{x}^i = 1$ for all $i \in \bar{f}, a \in A$. Clearly, $\tilde{x}$ is an LP-feasible solution and achieves the claimed objective. □

Lemma 3.2.2. *For every graph G,* $\mathrm{ILP}^D_{\mathrm{Base}}$ *admits an LP-feasible solution with objective value $\bar{f}_{>D}$. If G contains no closed walk of length at most D, this value is optimal.*

Proof. The first claim follows from the LP-feasible solution $\tilde{x}^a_i = 1/\bar{f}_{>D}$ and $\tilde{x}_i = 1$ for all $i \in \bar{f}_{>D}, a \in A$, and $\tilde{y} = 0$. When $\mathcal{C}_{\leq D}$ is empty, there are no y variables and the domains (3.4f) bound the objective from above, yielding the second claim. □

Lemma 3.2.3. *Model* $\mathrm{ILP}^{D+1}_{\mathrm{Base}}$ *is at least as strong as* $\mathrm{ILP}^D_{\mathrm{Base}}$ *for any $D \geq 2$.*

Proof. Observe that $\mathrm{ILP}^{D+1}_{\mathrm{Base}}$ generally contains more y- but fewer x-variables than $\mathrm{ILP}^D_{\mathrm{Base}}$. Consider an LP-feasible solution $(\hat{x}, \hat{y})$ for $\mathrm{ILP}^{D+1}_{\mathrm{Base}}$. We derive an LP-feasible solution $(\tilde{x}, \tilde{y})$ for $\mathrm{ILP}^D_{\mathrm{Base}}$ that achieves no smaller objective value. For notational simplicity, let $\hat{x}_i = \hat{x}^a_i \coloneqq 0$ for all $i > f_{>D+1}$ and $\beta \coloneqq \sum_{c \in \mathcal{C}_{=D+1}} \hat{y}_c$. For $\beta = 0$, already $(\hat{x}, \hat{y})$, when interpreted for $\mathrm{ILP}^D_{\mathrm{Base}}$, is LP-feasible. Assume $\beta > 0$. Let $\alpha \coloneqq \bar{f}_{>D} - \sum_{i=1}^{\bar{f}_{>D+1}} \hat{x}_i$ and $\beta_a \coloneqq \sum_{c \in \mathcal{C}_{=D+1}: a \in c} \hat{y}_c \ \forall a \in A$. From $\alpha < \beta$ it would follow that $\bar{f}_{>D} < \sum_{i=1} \bar{f}_{>D+1} \hat{x}_i + \beta$, a direct contradiction of constraint (3.4e) for $d = D$ in $\mathrm{ILP}^{D+1}_{\mathrm{Base}}$. Thus, $\alpha \geq \beta$. We define $(\tilde{x}, \tilde{y})$ by $\tilde{x}_i \coloneqq \hat{x}_i + (1 - \hat{x}_i)\beta/\alpha, \forall i \in [\bar{f}_{>D}]$; $\tilde{x}^a_i \coloneqq \hat{x}^a_i + (\tilde{x}_i - \hat{x}_i)\beta_a/\beta, \forall i \in [\bar{f}_{>D}], a \in A$; and $\tilde{y}_c \coloneqq \hat{y}_c, \forall c \in \mathcal{C}_{\leq D}$. The

objective value (3.4a) for $(\tilde{x}, \tilde{y})$ in $\mathrm{ILP}^D_{\mathrm{Base}}$ is

$$\begin{aligned}
\textstyle\sum_{i=1}^{\bar{f}_{>D}} \tilde{x}_i + \sum_{c\in\mathcal{C}_{\leq D}} \tilde{y}_c &= \textstyle\sum_{i=1}^{\bar{f}_{>D}} \big(\hat{x}_i + (1-\hat{x}_i)\beta/\alpha\big) + \sum_{c\in\mathcal{C}_{\leq D}} \hat{y}_c \\
\text{(by } x_i = 0 \ \forall i > \bar{f}_{D+1}\text{)} \quad &= \textstyle\sum_{i=1}^{\bar{f}_{>D+1}} \hat{x}_i + \beta/\alpha \cdot \big(\bar{f}_{>D} - \sum_{i=1}^{\bar{f}_{>D+1}} \hat{x}_i\big) + \sum_{c\in\mathcal{C}_{\leq D}} \hat{y}_c \\
\text{(by def. of } \alpha, \beta\text{)} \quad &= \textstyle\sum_{i=1}^{\bar{f}_{>D+1}} \hat{x}_i + \sum_{c\in\mathcal{C}_{\leq D+1}} \hat{y}_c,
\end{aligned}$$

i.e., equal to that of $(\hat{x}, \hat{y})$ in $\mathrm{ILP}^{D+1}_{\mathrm{Base}}$. Assuming constraint (3.4b) to be violated, we obtain $(D+1)\tilde{x}_i > (D+1)(\tilde{x}_i - \hat{x}_i) + \sum_{a\in A} \hat{x}^a_i$, since $\sum_{a\in A} \beta_a/\beta = D+1$. This implies $(D+1)\hat{x}_i > \sum_{a\in A} \hat{x}^a_i$, a violation of constraint (3.4b) already by $(\hat{x}, \hat{y})$. Let us show the feasibility of $(\tilde{x}, \tilde{y})$ w.r.t. constraints (3.4c) by expanding their left-hand side.

$$\begin{aligned}
\tilde{x}(a) + \tilde{y}(a) &= \textstyle\sum_{i=1}^{f_{>D}} \big(\hat{x}^a_i + (1-\hat{x}_i)\beta_a/\alpha\big) + \sum_{c\in\mathcal{C}_{\leq D}: a\in c} \hat{y}_c \\
\text{(by def. of } \alpha\text{)} \quad &= \textstyle\sum_{i=1}^{\bar{f}_{>D+1}} \hat{x}^a_i + \beta_a + \sum_{c\in\mathcal{C}_{\leq D}: a\in c} \hat{y}_c \\
\text{(by def. of } \beta_a\text{)} \quad &= \textstyle\sum_{i=1}^{\bar{f}_{>D+1}} \hat{x}^a_i + \sum_{c\in\mathcal{C}_{\leq D+1}: a\in c} \hat{y}_c \quad \text{(by feasibility of (3.4c) in } (\hat{x}, \hat{y})\text{)} = 1
\end{aligned}$$

Since $\mathcal{C}_{=D+1}$ contains only closed walks, we have $\sum_{u\in \mathrm{N}(v)} (\beta_{uv} - \beta_{vu}) = 0$ for all nodes v. Constraints (3.4d) hold, as we see by expanding their left-hand side:

$$\begin{aligned}
\textstyle\sum_{vu\in A} \tilde{x}^{vu}_i &= \textstyle\sum_{vu\in A} \hat{x}^{vu}_i + (\tilde{x}_i - \hat{x}_i)/\beta \cdot \sum_{vu\in A} \beta_{vu} \\
\text{(by (3.4d) in } \mathrm{ILP}^{D+1}_{\mathrm{Base}} \text{ and the above)} \quad &= \textstyle\sum_{uv\in A} \hat{x}^{uv}_i + (\tilde{x}_i - \hat{x}_i)/\beta \cdot \sum_{uv\in A} \beta_{uv} \\
&= \textstyle\sum_{uv\in A} \tilde{x}^{uv}_i
\end{aligned}$$

Constraints (3.4e) maintain their slack, as the first term increases by $\sum_{i=1}^{\bar{f}_{>D}} (\tilde{x}_i - \hat{x}_i) = \beta$ while the second decreases by β. Clearly, $\tilde{x}_i \geq \hat{x}_i$ and $\tilde{x}^a_i \geq \hat{x}^a_i$. By $\alpha \geq \beta$ we have $\tilde{x}_i \leq 1$. By (3.4c) we have $\hat{x}^a_i + \beta_a \leq 1$. Thus, $\tilde{x}^a_i > 1$ would imply $\tilde{x}_i - \hat{x}_i > \beta$. Clearly, we keep $0 \leq \tilde{y}_c \leq 1$. □

Theorem 3.2.4. *Model* $\mathrm{ILP}^{D+1}_{\mathrm{Base}}$ *is stronger than* $\mathrm{ILP}^D_{\mathrm{Base}}$ *for any* $D \geq 2$.

Proof. Restricting ourselves to dense graphs of girth greater than $D+1$, the claim immediately follows from Lemmas 3.2.1 to 3.2.3. An example of such graphs are the complete graphs on D nodes, where we subdivide each edge D times. They have girth $3(D+1)$ and are dense enough such that the respective bounds differ: $f_{>D} > f_{>D+1}$. We note that there are also dense graphs with high girth that do not allow any general preprocessing techniques. □

3.3 Additional Tuning

Here we describe various "add-ons" to the above models. We begin with a set of supplemental constraints that may be applied to several of these models. Let us discuss them in alphabetical order.

Arc-Face. We may require the below trivial constraints explicitly.

$$x_i \geq x_i^a \qquad \forall a \in A, i \in [\bar{f}] \tag{3.7}$$

Branching Rule. To facilitate the fast generation of strong primal bounds, we may initially restrict the solution space to explicitly modeled faces, e.g., by branching on

$$\textstyle\sum_{i \in \bar{f}_{>D}} x_i \stackrel{?}{=} 0. \tag{3.8}$$

deg 3-Model. There are only two possible rotations around any degree-3 node v. In $\mathrm{ILP}_{\mathrm{Pre}}$, this can be modeled by a single binary variable for v and alternative constraints, partially replacing (3.2a–3.2d), as discussed in [Bey+16]. The same holds for $\mathrm{ILP}^D_{\mathrm{Pre}}$, and we use this presumed improvement in our benchmarks.

For a node v with $\deg(v) = 3$, let $u_j \in \{u_0, u_1, u_2\}$ denote its jth neighbor. Here, operations on j are to be understood modulo 3. The only possible rotations (up to cyclic shifts) at v are $u_0u_1u_2$ (if $p_v = 1$) and $u_0u_2u_1$ (otherwise).

$$x_i^{vu_{k+1}} \geq x_i^{u_k v} + p_v - 1 \qquad \forall i \in [\bar{f}], v \in V : \deg(v) = 3, k \in [\![2]\!] \tag{3.9a}$$
$$x_i^{u_k v} \geq x_i^{vu_{k+1}} + p_v - 1 \qquad \forall i \in [\bar{f}], v \in V : \deg(v) = 3, k \in [\![2]\!] \tag{3.9b}$$
$$x_i^{vu_k} \geq x_i^{u_{k+1} v} - p_v \qquad \forall i \in [\bar{f}], v \in V : \deg(v) = 3, k \in [\![2]\!] \tag{3.9c}$$
$$x_i^{u_{k+1} v} \geq x_i^{vu_k} - p_v \qquad \forall i \in [\bar{f}], v \in V : \deg(v) = 3, k \in [\![2]\!] \tag{3.9d}$$

Long Faces. In several cases we can establish lower bounds on the length of faces modeled by the x-variables. Let $\mathrm{s}(v, w)$ denote the lengtthh of the shortest path between nodes v and w.

Lemma 3.3.1. *For any two arcs uv, wx that traverse a common face f, we have*

$$\mathrm{s}(v, w) + \mathrm{s}(x, u) + 2 \leq |f|.$$

Proof. Tracing any such face f yields a path from v to w that neither contains arc uv (it may contain arc vu) nor arc vx. Similarly, an arc-disjoint path

from x to u must exist in f. The total length of these paths is lower bounded by $\mathrm{s}(v,w)+\mathrm{s}(x,u)$. □

Lemma 3.3.2. *Any face with a singular edge contains at least eight arcs and this is tight.*

Proof. Let uv denote the edge that is traversed in both directions when tracing face f. If tracing f would additionally yield a path from u to v that does not contain uv, the tracing would similarly yield a path from v to u that does not contain vu. This contradicts the assumption since f would contain two oppositely directed, closed walks that form two separate faces. Hence, there exist two arc-disjoint closed walks on the boundary of f, one for each node u, v. Since there are no leaves, i.e., nodes of degree 1, in a biconnected graph, any subcycle in a face requires at least three arcs and the claim follows. Considering a genus-1 embedding of the K_4 we can see that it indeed contains such a face of length eight. □

Lemma 3.3.3. *Any face with a singular node contains at least six arcs and this is tight.*

Proof. If the face f also traverses an edge twice, the bound follows from Lemma 3.3.2. Otherwise, the doubly traversed node has at least four arcs in f, belonging to pairwise different edges, and hence four incident nodes. A closed walk on this $K_{1,4}$ requires at least two additional arcs and the claimed bound follows. A face of length six can be observed in a genus-1 embedding of the following graph: Take two copies of the K_5, remove one edge each, join the graphs by identifying two deg-3 nodes, and add a new edge between the remaining two deg-3 nodes. □

Let $\ell_{uv,wx} \coloneqq \max\{\mathrm{s}(v,w)+\mathrm{s}(x,u)+2, 6\cdot\mathbb{1}_{k=3}, 8\cdot\mathbb{1}_{k=2}\}$ with $k \coloneqq |\{u,v,w,x\}|$. Lemmas 3.3.1 and 3.3.3 yield the following constraints. When using them in our benchmarks, we separate them.

$$\ell_{uv,wx}(x_i^{uv}+x_i^{wx}-1) \le x(i,A) \quad \forall i \in [\bar{f}_{>D}],\ uv,wx \in A : uv \neq wx \tag{3.10}$$

Objective Parity. All above ILPs maximize the number f of faces and deduce γ via Euler's formula $n+f-m=2-2\gamma$. Thus, the parity of f is fixed. This gives room for improved bounding and cutting by the ILP solver. Using a new variable $z \in \mathbb{N}$ we may demand

$$(m-n \bmod 2)+2z = \textstyle\sum_{i=1}^{\bar{f}_{>D}} f_i + \sum_{c\in\mathcal{C}_{\le D}} y_c. \tag{3.11}$$

Symmetry Breaking. For ILP_{Pre}, it was observed in [Bey+16] that symmetry breaking does not seem to pay off. However, ILP_{Pre} only solves genus-1 instances in practice (see below). Symmetry breaking may hinder heuristics from identifying trivially optimal solutions, but may be beneficial for harder instances. The approach in [Bey+16] enforces that face i has at most as many arcs as face $i+1$. However, there are typically many faces with the same length in a low-genus embedding. We consider breaking symmetries by restricting the set of faces that may contain a given arc. Let $\prec$ denote an arbitrary but fixed order on the arc set A.

$$y(a) + x(\{1, ..., \ell\}, a) \geq 1 - x(\ell, \{a' \in A : a' \prec a\}) \quad \forall \ell \in [\bar{f}], a \in A \quad (3.12)$$

These constraints ensure that any arc is contained either in an explicitly modeled closed walk or in the lowest-indexed face that it can be placed into. For ILP_{Pre} and ILP_{Real}, i.e., when there are no explicitly modeled closed walks, we simply set $y(a) = 0$.

3.4 Experiments

All algorithms are implemented in `C++`, compiled with `gcc` 6.3.0, and use the OGDF (snapshot 2018-03-28) [Chi+13]. We use SCIP 6.0.0 [Gle+18] for solving ILPs, with CPLEX 12.8.0 as the underlying LP solver. Each computation uses a single physical core of a Xeon Gold 6134 CPU (3.2 GHz) with a memory speed of 2666 MHz. We employ a time limit of 10 minutes and a memory limit of 8 GB per computation. All instances and results, giving running time and genus (if solved), are available for download at `http://tcs.uos.de/research/min-genus`. In our experiments, we increase the parameter D—separately on each graph—until we obtain at least 1000 y-variables. As they are not required for integral solutions, we omit constraints (3.4e) by default.

Instance Sets. We consider the two established real-world sets North and Rome as well as a set of (artificial) Regular graphs. See Section 1.7.1 for general details on their size, relevance, and methods used for generation.

Discussion of SAT-based algorithms. In [Bey+16], the SAT-based approach was faster than the ILP-based one. However, we do not need to directly compare with it.

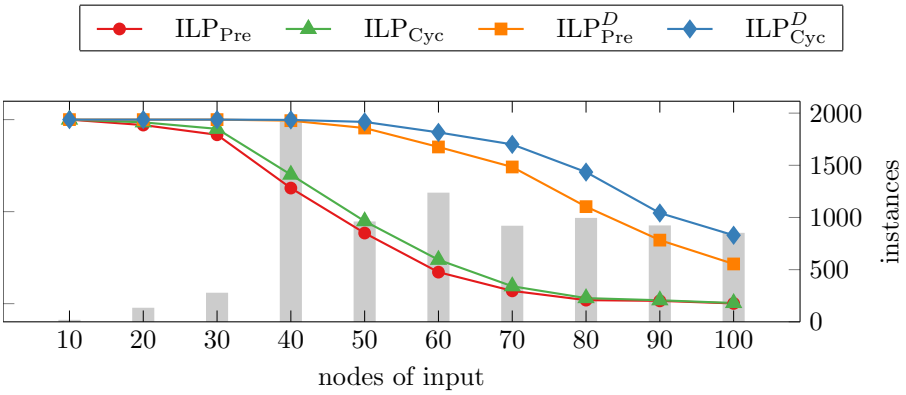
Both previous approaches solve only planar and toroidal instances in practice, i.e., those with genus at most one. Since the respective dual bound

is trivially given by planarity testing (and enforced in all previous models), the running time difference can be attributed to the SAT-solver quickly finding a satisfying solution. In contrast, standard primal heuristics of ILP-solvers are weaker, and the comparably time-consuming LP-relaxations are rarely profitable. However, w.r.t. success-rate, SAT is only marginally in the lead, if at all: on the ROME graphs, the ILP and SAT solve 2595 and 2667 instances, respectively. For NORTH, "the success-rates of both approaches are [...] comparable" [Bey+16].

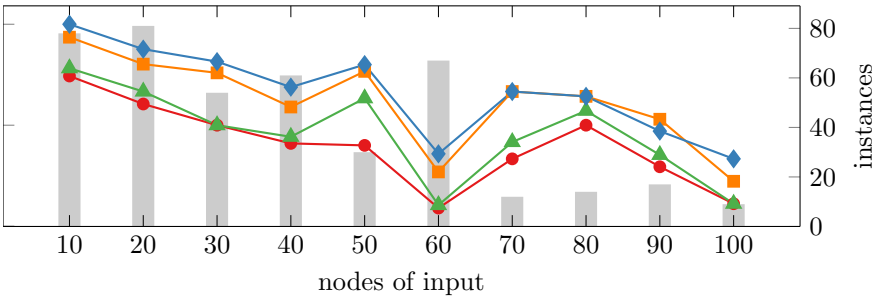
Since we can neither employ separation nor LP-relaxations in the setting of SAT-solvers, there also is no immediate way of using our strengthening results for SAT-based algorithms. We will see that the new ILP variants clearly dominate the SAT-based variant; e.g., we solve up to 6797 ROME instances.

Results. The experiments confirm that our new model is not only theoretically stronger but also better in practice: Using $\text{ILP}^D_{\text{Real}}$, we are now able to solve 82% instead of just 28% of the ROME graphs, cf. Table 3.1. Depending on the instance set, we achieve an average speed-up of factor 82 to 248. In [Bey+16], only graphs with genus 1 (and not all of them) could be solved. Surprisingly, and in contrast to the observations made in [Bey+16], the deg 3-model does not perform better than the respective base variant: SCIP's built-in preprocessing reduces the variable space to essentially the same dimension as obtained when manually applying the deg 3-model (while possibly retaining some additional information that helps in the solving process). Also somewhat to our surprise, none of the add-ons (3.8–3.12) pay off *reliably*.

Taking a closer look at the number of solved instances (Fig. 3.1 and Table 3.2), we see that—on average—$\text{ILP}^D_{\text{Real}}$ is superior to all other variants for any graph size. We now solve real-world instances with non-trivial dual bounds, i.e., when the genus is larger than 1, e.g., we have solved an instance with genus 7 from ROME and even a one with genus 21 from NORTH. We see very clearly, in particular on ROME, that we may order the models as ILP_{Pre}, ILP_{Real}, $\text{ILP}^D_{\text{Pre}}$, $\text{ILP}^D_{\text{Real}}$ by increasing success rate. This means that, independent on whether we apply the small-faces model extension or not, the realizability model is more successful than the predecessor model. The most progress, however, is achieved by activating the small-faces model extension $\text{ILP}^D_{\text{Base}}$. As the shapes of the success-rate curves demonstrate, it benefits both underlying models roughly equally. In particular, we see that even ILP_{Real}, like ILP_{Pre}, can only solve genus 1 instances (cf. also



(a) Solved ROME graphs by number of nodes.

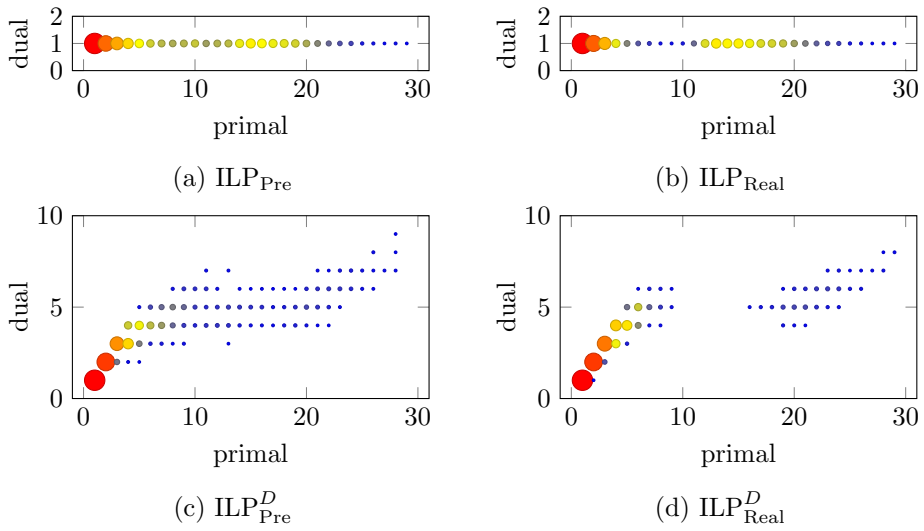


(b) Solved NORTH graphs by number of nodes.

Figure 3.1: Detailed success-rates of algorithms on established benchmark sets. We provide the relative number of solved instances over the number of nodes, clustered to the nearest multiple of 10. The gray bars denote the number of instances in each cluster.

Table 3.1: Average success-rate s and running time t on each instance set. Considering the running time, we restrict the instances to those solved by all variants. Of the graphs solved by $\mathrm{ILP}_{\mathrm{Pre}}$, all variants solved at least 94% (Rome), 99% (North), and 100% (Regular). Particularly, algorithms based on $\mathrm{ILP}^{D}_{\mathrm{Base}}$ always solved all of the instances solved by $\mathrm{ILP}_{\mathrm{Pre}}$.

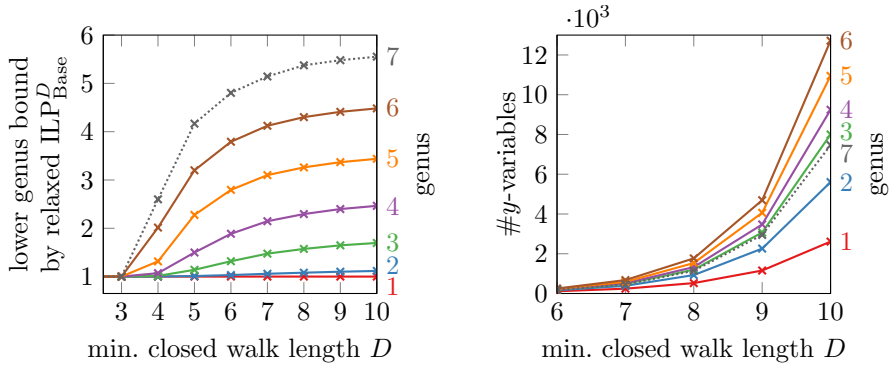
	Rome		North		Regular	
	s [%]	t [s]	s [%]	t [s]	s [%]	t [s]
$\mathrm{ILP}_{\mathrm{Pre}}$	27.79	190.23	45.86	139.09	5.26	58.29
$\mathrm{ILP}_{\mathrm{Pre}}$ + deg 3	27.94	186.31	47.52	111.50	5.26	58.53
$\mathrm{ILP}_{\mathrm{Real}}$	**31.85**	**70.57**	**50.83**	**25.44**	**5.53**	**11.33**
$\mathrm{ILP}^{D}_{\mathrm{Pre}}$	73.08	2.92	67.14	12.21	17.37	2.24
$\mathrm{ILP}^{D}_{\mathrm{Pre}}$ + deg 3	68.09	3.86	65.01	12.47	17.37	2.25
$\mathrm{ILP}^{D}_{\mathrm{Real}}$	**81.65**	**0.91**	**73.52**	**7.26**	**23.95**	**0.95**
$\mathrm{ILP}^{D}_{\mathrm{Real}}$ with add-ons						
branch rule (3.8)	76.41	0.86	73.05	7.29	21.05	0.80
all symmetries (3.12)	81.56	0.91	73.52	7.27	23.95	0.95
sepa. symmetries (3.12)	81.59	0.91	73.76	7.26	23.95	0.94
sepa. long faces (3.10)	81.16	0.95	72.81	7.27	22.89	0.81
all #faces cons. (3.4e)	81.57	**0.75**	**74.00**	6.79	23.68	0.86
sepa. #faces cons. (3.4e)	81.60	1.13	**74.00**	7.02	23.95	1.05
parity model (3.11)	**82.33**	1.25	73.29	**1.11**	22.89	1.68
all arc-face cons. (3.7)	75.43	0.98	71.39	7.35	18.95	**0.71**
sepa. arc-face cons. (3.7)	81.71	1.48	73.76	7.28	23.42	0.76
(3.11), sepa. (3.4e,3.7)	**82.40**	1.37	74.00	1.20	22.63	1.65
(3.8,3.11), sepa. (3.4e,3.7)	78.07	1.19	**75.65**	1.18	21.58	1.62



(a) $\mathrm{ILP}_{\mathrm{Pre}}$ (b) $\mathrm{ILP}_{\mathrm{Real}}$

(c) $\mathrm{ILP}^{D}_{\mathrm{Pre}}$ (d) $\mathrm{ILP}^{D}_{\mathrm{Real}}$

Figure 3.2: Final primal vs. dual bounds on the genus, generated by algorithmic variants on ROME (without any add-ons). Color and size indicate the number of instances with the respective bounds. We note that these bounds do not apply to the values of the formal objective value, i.e., the number of attained faces, but to the genus, which allows a more sensible comparison. Note that without the small faces extension, neither $\mathrm{ILP}_{\mathrm{Pre}}$ nor $\mathrm{ILP}_{\mathrm{Real}}$ obtains lower bounds > 1 (i.e., only trivial ones).



(a) genus bound of LP-relaxation (b) number of generated y-variables

Figure 3.3: Average values of relaxed $\mathrm{ILP}^{D}_{\mathrm{Base}}$ on the solved ROME graphs, depending on the maximum length D of explicitly modeled cycles and the genus of the graph. Note that we only solved one instance with genus 7 on ROME.

Table 3.2: Number of solved instances in the REGULAR set for selected variants without add-ons.

# nodes	10		20			30		≥ 50
node degree	4	6	4	6	10	4	6-20	4–40
ILP_{Pre}	20	0	0	0	0	0	0	0
ILP_{Real}	20	1	0	0	0	0	0	0
$\text{ILP}^{D}_{\text{Pre}}$	20	20	20	0	0	6	0	0
$\text{ILP}^{D}_{\text{Real}}$	20	20	20	18	0	13	0	0

Fig. 3.2 which shows the final bounds of our core variants on ROME). This is in accordance with Lemma 3.2.1, i.e., that the LP-relaxation of ILP_{Base} always yields value $\bar{f}$. Nonetheless, the success-rates 46% and 51% on NORTH for ILP_{Pre} and ILP_{Real}, respectively, demonstrate that ILP_{Pre} is far from solving all toroidal instances. More complex instances require the small-faces extension $\text{ILP}^{D}_{\text{Base}}$. This is also reflected by the root relaxations of $\text{ILP}^{D}_{\text{Base}}$ for different values of D, cf. Fig. 3.3. Consistent with theory, increasing the minimum length D leads to stronger LP-relaxations also in practice, but may drastically increase the number of variables. Interestingly, generating only triangles, i.e., $D = 3$, yields only a very slight increase on the average dual bound compared to ILP_{Base} on ROME, possibly caused by the graphs' sparsity.

Genera in Graph Theory. Our new approach allows us to confirm results from literature, all with non-trivial dual bounds: In 2015, the circulants of genus at most 2 were characterized [CG15]. Thereby, the authors need to show that 12 specific graphs have genus at least 3. For these arguments alone, they require about nine pages, supplemented by several hours of computation. Using $\text{ILP}^{D}_{\text{Real}}$, we are able to confirm these results (and compute the respective genera) in a matter of seconds without employing any graph-specific theory. Before, using ILP_{Pre}, the arguably hardest case $C_{11}(1, 2, 4)$ required 180 hours [Bey+16]. In 2005, a full paper was dedicated to showing that the Gray graph has genus 7 [MPW05]. Our tool confirms this result within 42 hours. Similarly, we confirm a result from 1989 [BRS89] in 250 seconds: the group that is the semi-direct product of $\mathbb{Z}_9$ with $\mathbb{Z}_3$ has genus 4 (the genus of group Γ is the smallest genus of a Caley graph of Γ).

Conclusion. We have presented novel ILP models for the graph genus problem, proved their theoretical strength, the existence of a hierarchy of ever stronger LP-relaxations, and positively evaluated them in practice, e.g., by solving 82% instead of the previous 28% of the ROME instances. We are now able to solve real-world instances with genera up to 21. This is in stark contrast to the previous models that—on the same set of instances—succeeded only on toroidal graphs.

It remains open whether even stronger models can be found by a more careful examination of the face structure. What additional properties of the embedding may be modeled? Is it possible to better exploit singular nodes or edges, particularly when they are adjacent? Further, we expect that our algorithms would benefit from strong primal heuristics but we are not aware of any general such algorithms. Currently, optimal dual bounds are often identified long before an optimal solution is found.

Algorithm by G. Brinkmann. Just recently, a new and purely combinatorial algorithm by Gunnar Brinkmann emerged (implemented in pure `C`). His implementation is able to beat the performance of our fastest ILP model on most instances [Bri20]. The algorithm is based around a B&B procedure that iteratively embeds all edges (each in every possible way). One core aspect of this procedure is to identify edges that cannot be embedded without increasing the genus of the surface.

It remains open whether an amalgamation of both ideas may be used to design even faster algorithms.

Chapter 4

Crossing Number

Cycle (*history*). A possibly repeating interval of space or time in which one set of events or phenomena is completed.

In this chapter, we consider the notoriously hard crossing number problem. As opposed to the previous problems, we do not directly consider stronger or practically better suited ILP models. In fact, it seems surprisingly hard to apply the cycle models (that we successfully used for skewness and genus) in the context of crossing number ILPs: They require binary decision variables that each represent a potential short face in the resulting embedding. However, in the context of crossing number, short faces are not just cycles (or closed walks) in the graph. For example, there may be a triangular face whose boundary consists of three edge segments and three crossings without any nodes. As such, it seems impractical to model all potential short faces. Instead, in Section 4.1, we improve upon the strongest algorithms for general crossing number in a different direction by proposing an ILP-based proof system that automatically verifies the complex computations required to solve the state-of-the-art models. Its primary benefit is to automatically offer an easily verifiable proof for the crossing number of any (reasonably large) given graph which we will use in the following section. While discussing the well-known underlying ILP model, we will also introduce a new and simpler column generation scheme. Secondly, in Section 4.2, we study the minimal obstructions for low crossing number. More precisely, we consider a conjecture regarding nodes of arbitrarily high degree in such obstructions and sharpen its restricted validity by a constructive proof. Clearly, being able to quickly identify (some of) these obstructions immediately yields separation

procedures for potentially stronger ILP constraints. Finally, in Section 4.3, we briefly explore the complexity of adding an edge to a simple drawing which is, for example, of interest when experimentally exploring the—still widely open—crossing number of the complete graph.

4.1 Proof System

A typical corner stone of (integral) mathematical programming formulations for combinatorial optimization problems is that solving the formulation constitutes a *proof* for the optimality of the obtained solution. While this is true in theory, it is not clear, per se, that this actually transfers into practice, as many factors may influence or invalidate the program's outcome: the probably most prominent ones are hidden bugs in the software or numerical instabilities. E.g., [App+09] discusses the problems and challenges of proving the correct computation of an optimal TSP tour for *one* specific instance; we are interested in a system to deduce proofs automatically, without any human interaction.

Hence, while there exist many successful formulations, e.g., as ILPs, to many important graph-theoretic (optimization) problems, successful computations are generally not considered to be proofs accepted by the graph theory community.

We aim at bridging this gap for the well-known *crossing number problem*, which is arguably one of the most prominent and notorious problems in topological graph theory. We propose a system to extract a *simply verifiable* proof from a successful ILP computation, which can be accepted by graph theorists: It is shielded against ill-effects based on the software realization of the mathematical model. To understand the proof, the graph theorist does *not* have to have a deeper understanding of mathematical programming nor the required implementations; she only has to understand the mathematical model (and possibly a simple proof verification program, designed to be readable and checkable by non-experts).

When describing the proof system, we will also pinpoint the generally necessary differences in the formulation-, algorithm-, and software-design between the typical goal of obtaining a strong and fast solver and the goal of obtaining a system for easily understandable proofs. Crossing number formulations are particularly interesting in that respect, as their implementations need to combine a diverse set of different tools (branch-and-cut-and-prize with exact and heuristic constraint separation, column generation with non-standard bounding schemes, intricate heuristics for primal bounds, etc.) to

obtain a system that can solve realistically-sized instances. This inevitably leads to a software too complex to directly check against all possible bugs. As such, even though we focus on the crossing number problem, our system may serve as a showcase of how to obtain trustworthy mathematical programming based proof systems for graph-theoretic problems whose formulations do not allow easily checkable implementations.

We recall that the only known practical method to obtain the crossing number of a given graph is based in integer linear programs [Buc+08; CGM10; CMB08], based on two different modeling ideas to be described below. However, the models contain both too many variables and too many constraints to be solved directly via off-the-shelf techniques, and require a lot of (bug-prone) implementation effort. Even if implemented correctly, solving such ILPs can be error-prone due to numerical instabilities. This constitutes a problem for graph theorists, interested in utilizing the computed crossing number in a proof.

Basics. When considering the crossing number of a graph, it is well-known that it suffices to consider *simple* (also called *good*) drawings: no edge crosses itself; adjacent edges do not cross; each pair of edges crosses at most once; and no three edges cross in a common point. Considering such an optimal drawing of G, we can obtain a *planarization* of G, which is the graph arising from G when replacing each crossing with a new *dummy* vertex of degree 4. We may speak of a *partial planarization* if we substitute only certain crossings via new vertices such that the obtained graph possibly remains non-planar.

4.1.1 Known ILP Models

All known ILP models have a common core idea; they differ in how to handle the arising *realizability problem*, described below. We will only describe the formulation on a level necessary to comprehend the proof system (and the design decisions that lead to it). For a more detailed and formal description see the individual publications [Buc+08; CGM10; CMB08] or the full compilation in [Chi08].

Let $G = (V, E)$ denote the given simple and undirected graph for which to compute $cr(G)$, and let $\mathcal{CP} \coloneqq \big\{\{e, f\} \subseteq E : e \cap f = \varnothing\big\}$ be the set of edge pairs that potentially cross in a simple drawing of G. Consider a binary variable x_c for each $c \in \mathcal{CP}$ that is 1 if and only if the edge pair crosses. This gives the objective function

$$\min \textstyle\sum_{\{e,f\} \in \mathcal{CP}} w(e) \cdot w(f) \cdot x_{\{e,f\}} \tag{4.1}$$

where $w(e)$ denotes the (integral) weight of edge e. For usual (i.e., unweighted) graphs we have $w(e) = 1$ for all $e \in E$. In our implementation, we first preprocess G to obtain a smaller but integrally-weighted graph with the same crossing number [CG09].

To guarantee feasible solutions, we may be tempted to introduce the following *Kuratowski constraints*. Let $K \subseteq E$ be an edge subset forming a Kuratowski subdivision; we want to ensure that each such K gives rise to at least one crossing. We say that a crossing $c \in (K^{\{2\}} \cap \mathcal{CP})$ is *planarizing* if the graph obtained from K by realizing c via a dummy vertex is planar. Clearly, a crossing is planarizing if and only if the crossing edges do *not* belong to adjacent (or identical) Kuratowski paths. Let $\mathcal{CP}(K)$ be the set of planarizing crossings for $K \subseteq E$. We may use the following constraints:

$$\sum_{c \in \mathcal{CP}(K)} x_c \geq 1 \quad \forall \text{ Kuratowski subdivisions } K \text{ in } G.$$

While the above constraints form facets of the crossing number polytope [Chi11], they do not suffice to guarantee feasibility: On the one hand, we also have to consider Kuratowski subdivisions that only appear in partial planarizations due to dummy vertices. On the other hand, even those do not suffice: Let $R \subseteq \mathcal{CP}$ be edge pairs that are supposed to cross. The *realizability problem* is to decide whether there exists a drawing of G such that only the edge pairs R cross. Interestingly, even this seemingly simpler problem is still NP-hard for general graphs [Kra91]. Hence, our simple set of x-variables cannot suffice to describe the crossing number polytope. The key problem is that when two edges, say f and g, both cross an edge e, the *order* of these two crossings along e is of central importance and cannot be deduced in polynomial time (unless $\mathrm{P} = \mathrm{NP}$).

There are two approaches to tackle this problem. Both lead to a variable increase that, although polynomial, makes the models intractable in practice unless a dynamic column generation scheme is used. Assume in the following that we assign an arbitrary but fixed direction to each edge.

Subdivision-based exact crossing minimization (SECM). Let $G^{[\ell]}$ be the graph obtained from G by splitting each edge into a chain of $\ell \in \mathbb{N}^+$ edges (henceforth called segments). Instead of directly using the above model on G, we consider $G^{[\ell]}$ instead. We observe that the corresponding set $\mathcal{CP}^{[\ell]}$ will not need to contain edge pairs (segment pairs, in fact) where both segments belong to the same original edge in G (the underlying G does not require self-crossings).

On $G^{[\ell]}$, we search for the smallest number of crossings under the restriction that each segment is involved in at most one crossing. This restriction is

trivial to ensure via linear constraints (see later for details). The so-restricted crossing number is often called the *simple* crossing number, even though it is still NP-hard to decide. There can be at most $\chi := \min\{cr(G), |E|-1\}$ crossings on an edge in the optimal drawing of G. Hence, we may use any upper bound on χ as ℓ to ensure that the optimal solution to the restricted crossing number problem on $G^{[\ell]}$ induces an optimal solution for the usual crossing number on G. Since there are instances where an edge e needs to be crossed by $\Omega(|E|)$ many other edges in the optimal solution, our transformation may increase the number of variables $\Theta(|E|^2)$-fold.

The benefit of considering the simple crossing number is that the realizability problem becomes linear time solvable. We say a subset $R \subset \mathcal{CP}^{[\ell]}$ is *simple* if each segment occurs at most once over all segment pairs in R (i.e., it is a potential solution to the simple crossing number). For such an R, its corresponding (partial) planarization $P(R)$—obtained by substituting the crossings R in G with dummy vertices—is hence unique. We have R realizable if and only if $P(R)$ is planar.

Finally, we can ensure feasible solutions using more general Kuratowski-constraints. Let $\mathcal{K}(R)$ be the set of all Kuratowski subdivisions in $P(R)$. Each subdivision is specified by its edge set. Clearly, if the crossings R are part of the solution, each Kuratowski subdivision in $\mathcal{K}(R)$ will require at least one crossing. We have:

$$\sum_{c\in\mathcal{CP}(K)} x_c \geq 1 - \sum_{c\in R}(1-x_c) \quad \forall \text{ simple } R \subseteq \mathcal{CP}, K \in \mathcal{K}(R) \qquad (4.2)$$

In order to prove a lower bound of the crossing number, it suffices to understand that the constraints in the above model need to hold for any feasible solution. We do not need to argue about sufficiency (see also property $\langle 2\rangle$ below).

Ordering-based exact crossing minimization (OECM). The alternative formulation [CMB08] introduces *linear ordering variables* to resolve the order of crossings along each edge without subdividing the input graph. This has some advantages regarding performance, e.g., since every Kuratowski constraint in this model can cover more than just one specific partial planarization. Technically, these linear orderings are modeled using $\Theta(|E|^3)$ additional variables that are linked to the crossing variables using several constraint classes. Overall, it has to be observed that the OECM model, while offering superior performance, is much harder to understand and requires even more technically intricate column generation schemes, book-keeping, and subalgorithms, compared to SECM.

4.1.2 Design of Proof System

Without a proof system, one would need to check the ILP algorithms for correctness. All SECM and OECM implementations known to the authors are intertwined with the Open Graph Drawing Framework (OGDF) [Chi+13] and heavily utilize ABACUS [JT00]; they are all written in C++. The core of both algorithms roughly spans across 8,000 Lines of Code (LOC) while the OGDF amounts to a total of about 170,000 LOC. Already the main code paths of the research code are hard to comprehend without intricate knowledge of the algorithms. Furthermore, the programs use sophisticated column generation routines, requiring complex book-keeping and special constraint liftings to prevent a decrease of the lower bound when adding variables. Tracking variables and constraints over an entire algorithm is disproportionately harder than simply verifying each B&B leaf. Furthermore, there are several possibilities for hidden bugs due to numerical instabilities that may arise without any means of detection, or hidden buffer overruns when generating atypically many variables or constraints in one pass.

All these facts make a formal verification of the main algorithms intractable in practice. For comparison, our proof system proposes a verification procedure (written in Java) of less than 1,000 LOC (including rich documentation and comments), with a virtually complete test coverage.

In our context, a *proof* consists of three parts:

- a mathematical model (in our case the ILP formulation),
- a *witness* of the dual bound, and
- a primal *solution*, matching the above dual bound.

The first is independent of the specific instance but needs to be understood only once for a specific problem domain (crossing number, in our case). The latter two are instance-dependent. We want to make the proof as easily digestible by pure graph theorists as possible. To understand the proof it must be sufficient to understand the following:

⟨1⟩ *Solution.* One needs to be able to check the feasibility of a primal solution and evaluate its objective value. In our case, one needs to be able to recognize a feasible planarization of G, and to count the number of dummy vertices.

⟨2⟩ *Feasibility of the mathematical model.* It is only necessary to understand that all described constraints of the formulation are feasible; one need not concern oneself with understanding why the model is sufficient. Generally, it should be understood that any optimal fractional solution w.r.t. a

subset of the constraints gives a feasible dual bound (in our case: lower bound).

⟨3⟩ *Witness format.* The information contained in the witness that is required to verify the dual bound.

⟨4⟩ *Verification procedure.* The steps required to verify the dual bound claimed by the witness.

An ILP-based *proof system* should consist of two major components: the proof generation and the verification procedure. The arbitrarily complex proof generation produces a witness for the optimality of the primal solution. This witness contains all information required to create *relaxed* integer linear programs, i.e., LPs, for several *subcases* (the leaves in the B&B tree), all of which yield the dual bound. B&B leaves naturally resemble an easily verifiable case distinction, as used ubiquitously in graph-theoretic proofs.

The verification of the witness could *theoretically* be done by hand. It follows from the nature of NP-complete problems that (unless $\mathrm{P} = \mathrm{NP}$) there *cannot* be a really "simple" proof for the dual bound in general. If we want a simple-to-check witness for the dual bound (which is, most importantly, checkable in polynomial time w.r.t. its size), we *have* to live with the fact that the witness' size can grow exponentially with the size of G. In most cases, this sheer size will require us to introduce an—algorithmically very simple—computer-based verification procedure. Most importantly, the verification procedure only needs to check that the subcases described in the witness form LPs that are subsets of the underlying mathematical model. It requires no knowledge about the generation of constraints, variables, or branches.

We can summarize the general design goals for an ILP-based proof system:

G1. *Simplicity of model.* There should be few classes of constraints and variables. Comprehensibility outweighs performance as long as the proof procedure is still "fast enough".

G2. *Column generation.* Column generation should only be used if ultimately required. The variable subsets need to be as simple as possible.

G3. *LP-solver flexibility/provability.* The LP-solver used during verification should be easily interchangeable or self-proving.

G4. *Few Branches.* Superfluous branching decisions should be eliminated from the witness to keep it small. This can, e.g., be achieved by starting the extraction with a supposedly optimal primal bound, see below.

G5. *Human-readability.* One should be able to investigate certain aspects of the proof by hand. To achieve this, we may allow redundancy as long as conflicts are detected easily by the verifier.

G6. *Standalone verification.* The verifier must *not* share any resources (most importantly code fragments) with the extraction procedure. The witness and the primal solution constitute the sole interchange of information between generation and verification.

G7. *Coding standards.* Adhering to well-established coding standards when implementing the verifier increases readability. Likewise, an established programming language should be used. The verification procedure should be described in detail such that one might re-implement it easily.

Although the OECM formulation offers better performance than SECM in practice, goal G1 lets us favor the comparably much easier to understand SECM formulation. It also allows us to sacrifice more constraints classes to adhere to G1. Concerning G2, both formulations require complex column generation schemes to be feasible in practice. However, as we will describe below, SECM allows us to propose a new column generation scheme that is considerably simpler than any of the previously published ones for either of the two formulations, while increasing the running time and number of variables only mildly.

Algorithm 4 gives an outline of the proof generation. In order to obtain a small proof (G4), we solve the crossing number problem *twice*: First, we use the fastest OECM variant together with strong upper bound heuristics to obtain the presumably optimal solution. This procedure can be seen as a black box, as we are only interested in the fact that the solution gives an upper bound—we can check the feasibility of the primal solution straightforwardly. If our proof generation succeeds, this is the solution that is used as part of the overall proof. Now having this primal bound, we can start our modified SECM formulation (see below for details) without any primal heuristics and ask for a solution strictly better (at least one crossing less) than the obtained upper bound. From this second ILP run, we can extract all required information for each B&B leaf, to reconstruct each linear program that yields a lower bound on the number of crossings restricted to the solution space spanned by the branch. In general, the set of variables and constraints differ for any two leaves.

We observe that if OECM falsely returned a solution that is *not* optimal (e.g., due to a hidden bug), we may already detect this now as SECM's dual bound does not match our upper bound.

Algorithm 4: Proof generation

Input: graph G `// we are interested in` $cr(G)$

`// claim optimal planarization` P `with objective value` $\bar{c}$

1 $P, \bar{c} \leftarrow$ OECM(G)

2 verify(P) `// ensure the solution` P `is feasible`

`// generate a witness` W `for the lower bound, i.e.,`
`// prove that` $\bar{c} - 1$ `is infeasible to achieve`

3 $W \leftarrow$ modified-SECM(G, $\bar{c}$)

4 verify(W) `// use standalone verifier to confirm` W

5 print P and W `// output the proof`

4.1.3 Modified-SECM

There are two known column generation schemes for SECM [CGM10]. The *algebraic pricing*, based on the standard Dantzig-Wolfe decomposition theory, performs relatively weak and uses a quite unstructured variable subset. The more efficient *combinatorial* column generation scheme (denoted as *sparse* column generation in the following) can decide upon the addition of variables in a purely combinatorial fashion, and requires the fewest active variables in general. However, the required variable subset structure (and hence the reasoning for its sufficiency) is too complicated to easily describe and comprehend for the purpose of a graph-theoretic proof. We hence propose a new column generation scheme—called *homogeneous* in the following—by means of describing an SECM variant that is slightly modified compared to the model described above.

Instead of a simple number, let $\ell\colon E \to \mathbb{N}$ be a mapping describing the *expansion status* of G, i.e., we consider each edge $e \in E$ of G to be subdivided into $\ell(e)$ segments. We define our ILP using the resulting graph G^ℓ. For notational simplicity, let $e_1, e_2, ..., e_{\ell(e)}$ denote the segments of an original edge $e \in E$. As before, the new graph induces a set of segment pairs $\mathcal{CP}^\ell$ that may cross in an optimal solution. We explicitly allow (and expect) values $\ell(e)$ to be smaller than the upper bound of crossings over e. To this end, we allow at most one crossing over each segment *except for the first segment of each edge*: it may be crossed an arbitrary number of times. In general, this could lead to problems with testing realizability. However, this is of no concern to us, as we only require that our model is *feasible*, i.e., it

allows to describe an optimal solution (see ⟨2⟩). This is trivially the case in this setting (already when $\ell(e) = 1$ for all $e \in E$).

Symmetric solutions increase our set of subcases in the proof, often drastically: To counter this, we can require w.l.o.g. that the crossings over an original edge may be *aligned* in such a way that there is only a crossing on a segment if there also is a crossing on the previous one, cf. (4.3) and (4.4). The first segment is typically not part of this alignment scheme, as it allows multiple crossings. However, observe that given an upper bound $\bar{c}$ on $cr(G)$, any edge $e = \{u, v\} \in E$ may be crossed at most $u_e := \min\{\bar{c}, |E| + 1 - \deg(u) - \deg(v)\}$ times in the optimum solution. If an edge is fully expanded, i.e., $\ell(e) = u_e$, we do allow at most one crossing also over the first segment; to avoid symmetries we may further assume to have less crossings on the first than on the last segment (4.5). The following constraints establish these segment properties. They, together with the objective function (4.1) and the Kuratowski constraints (4.2) (both applied to the set $\mathcal{CP}^\ell$), form our full mathematical model.

$$\sum_{c \in \mathcal{CP}^\ell : e_2 \in c} x_c \leq 1 \qquad \forall e \in E \tag{4.3}$$

$$\sum_{c \in \mathcal{CP}^\ell : e_i \in c} x_c \leq \sum_{c \in \mathcal{CP}^\ell : e_{i-1} \in c} x_c \qquad \forall e \in E, 3 \leq i \leq \ell(e) \tag{4.4}$$

$$\sum_{\{e_1, f\} \in \mathcal{CP}^\ell} x_{\{e_1,f\}} \leq \sum_{\{e_{\ell(e)}, f\} \in \mathcal{CP}^\ell} x_{\{e_{\ell(e)},f\}} \qquad \forall e \in E : \ell(e) = u_e \tag{4.5}$$

Remark (Irrelevant to understanding the proof). As in SECM, Kuratowski constraints are separated via a procedure based on testing a rounded solution S for planarity. An effective column generation similar to [Buc+08] is achieved by starting with a unit vector ℓ and incrementing $\ell(e)$ whenever there are at least 2 crossings on e_1 in S (i.e., the realization problem cannot be solved uniquely).

4.1.4 Verification Procedure

Finally, we can focus on the actual verification steps necessary to prove the lower bound obtained by the above Modified-SECM model. Algorithm 5 gives an overview.

Branch Coverage. We need to make sure that the entire solution space is covered (cf. line 1 of Algorithm 5). Therefore, we consider the variable fixings in all subcases. Since we only branch on single variables, we iteratively merge two subcases that differ by the assignment of a single variable, giving a more general subcase without this variable being fixed at all. In a valid

Algorithm 5: Proof verification

Input: graph G, subcases $\mathcal{L}$ (= B&B leaves), and claimed lower bound $b \in \mathbb{N}_+$

1 **assert** allBranchesAreCovered($\mathcal{L}$)
2 **foreach** $\nu \in \mathcal{L}$ **do**
3 $\quad \ell \leftarrow$ expansion status at ν
4 $\quad \mathcal{K} \leftarrow$ set of Kuratowski subdivisions observed at ν
5 $\quad$ **foreach** $C \in \mathcal{K}$ **do**
6 $\quad\quad$ **assert** isKuratowski(G^ℓ, C)
7 $\quad P \leftarrow$ generateLinearProgram($G, \mathcal{K}$)
8 $\quad$ **assert** lpsolve(P) $> b - 1$

proof, this procedure must end with a single subcase, which does not have any variable fixings at all.

The following pseudo-code shows how to algorithmically verify that the subcases span the whole solution space. Here, we consider each subcase ν to be a set of tuples from $\mathcal{CP}^\ell \times \{0, 1\}$, i.e., a set of segment pairs that are specified to either cross (1) or not (0). Segment pairs not listed in ν are free to do either. Let $\triangle$ denote the symmetric difference.

Algorithm 6: Coverage verification

Input: subcases $\mathcal{L}$ (see text)

1 **while** $\exists \mu, \nu \in \mathcal{L}$ *with* $\exists c \in \mathcal{CP}^\ell : \mu \triangle \nu = \{(c, 0), (c, 1)\}$ **do**
2 $\quad L \leftarrow \{\mu \cap \nu\} \cup L \setminus \{\mu, \nu\}$
3 **assert** $\mathcal{L} = \{\varnothing\}$

Kuratowski Constraints. For each subcase (cf. line 2 of Algorithm 5), we need to check that only feasible constraints are considered. A Kuratowski constraint for a subgraph K that is not a Kuratowski subdivision would be an error. It would enforce a crossing that may not be necessary in the optimal solution. For each subcase, our witness explicitly stores each used Kuratowski subgraph K, together with the required crossings (R) that need to exist for K to arise[1]. More specifically, K is stored by means of Kuratowski paths $p_1^K, ..., p_k^K$. This storage pattern allows for a simpler verification (see

[1]While constraints *could* be stored without explicitly stating R, this would decrease the readability of the proof and the verification procedure, cf. G5.

below) than a general Kuratowski verification routine as, e.g., described in [NRM14].

To verify that K is really a Kuratowski subdivision (cf. line 6 of Algorithm 5), we first check whether each p_i^K is in fact a valid path (only exploiting crossings of R, if any). Then, we check that all paths are pairwise internally disjoint (i.e., they are disjoint except for possibly common start/end vertices). Finally, the set of nodes that constitute the start or end of all Kuratowski paths is collected. The size of this set (5, 6) and the number of Kuratowski paths (10, 9) is verified according to the type of the subdivision (K_5, $K_{3,3}$, respectively). The structural verification of a K_5 subdivision simply checks whether all 5 nodes are directly connected to one another via Kuratowski paths. For a $K_{3,3}$, we perform a two-coloring (interpreting the paths as edges); each of the 6 nodes must be connected to exactly 3 distinct nodes of the opposite color.

Lower Bound. Finally, we need to verify the lower bound for each subcase. We can trivially generate a linear program (no integrality constraints) according to our model (cf. lines 7–8 of Algorithm 5). From the expansion status ℓ we can construct $\mathcal{CP}^\ell$, the objective function (4.1) and the segment-oriented constraints (4.3) to (4.5). For each (already verified) Kuratowski subdivision considered in the subcase, we generate the corresponding constraint (4.2).

By writing this LP in a standard file, we can use *any* LP-solver (or multiple, to gain confidence) to verify that the LP's solution value is strictly larger than $\bar{c}-1$. For a more formal proof, we may check the basis of the final tableau/the dual solution to verify the lower bound and/or use self-proving LP solvers [App+07; Dhi+03].

4.1.5 Practice and Experiments

Web-Service. Already prior to our proof system, we offered a (free) web-service to compute the crossing number of an uploaded graph (see `http://crossings.uos.de`). Over the last years, it has been used as a tool by several research groups worldwide, to help validate or falsify crossing number conjectures and ideas. We collected the thereby uploaded instances. However, the web-service would (formally) only give a primal solution, together with the assertion that this *should* be the optimum. Now, we relaunched the web-service to also hand out the formal proof. The user can download the stand-alone Java verification program, check it, and use it to verify her proof independently.

Experimental Evaluation. To determine the applicability of the proof system, we tested the algorithms on three benchmark sets: the 1277 NORTH graphs, the 3110 non-trivial ROME graphs, and 145 non-planar graphs, (`http://crossings.uos.de/instances`) collected by our crossing number web-service. See Section 1.7.1 for details on the former two sets.

All experiments were conducted using an Intel Xeon E5-2430 v2, 2.50 GHz with 192 GB RAM running on Debian 8. We compiled with g++ 4.9.0 (64bit, -O3), used CPLEX 12.6.0 as the backend LP-solver, and applied a time-limit of 60 minutes for each computation. All algorithms except the verifier are implemented as part of the OGDF (using ABACUS as the ILP-framework). We compare the *sparse* to the newly introduced *homogeneous* column generation scheme, to understand the running time costs of the simpler but supposedly weaker column generation scheme. For both schemes we consider the cases whether we start with a *tight* upper bound (the optimum) or not; the former is the setting that we typically use within our proof system. Fig. 4.1 summarizes the results. While *tight homogeneous* requires more time than *tight sparse* on larger instances, it is still faster than *sparse* without a tight upper bound. On all instances with crossing number at most 22, *homogeneous* is at most 5 times slower than *sparse* (for both upper bound modes, considering only those instances solved by both schemes). Using the tight upper bound reduces the running time to about 30% on average for *homogeneous*. Out of all 3393 instances solved by *tight sparse*, only 11 could not also be solved by *tight homogeneous* in time. Thus, we conclude that the increase in running time due to the simpler column generation is reasonable in practice. The running time of the verification procedure is negligibly small.

Conclusion. We considered the problem of bridging the gap between "provably optimal" solutions obtained via mathematical programming and the demands on a verifiable formal proof. To this end, we laid out the general central design goals and steps to turn a mathematical program into a proof system.

We combined this with a showcase of how to automatically obtain a verifiable proof for the graph-theoretic crossing number problem, whose known ILP implementations are far from being formally checkable. To this end, we also introduced a novel column generation scheme for the problem's model, which, while much simpler, is still very effective in practice. The final proof system is available online for free academic use.

Since the relaunch of this tool roughly five years ago, we collected about

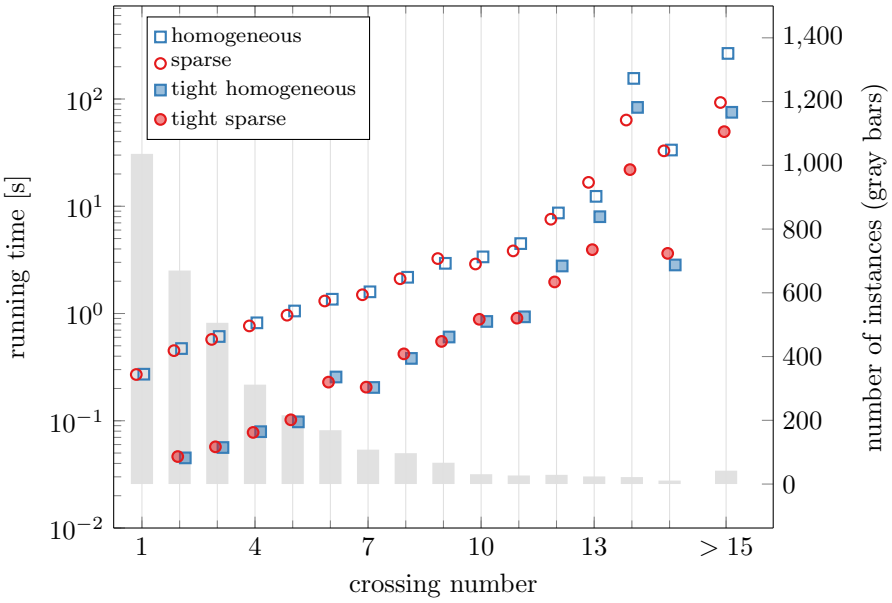


Figure 4.1: Running time of different SECM variants (see text). The tight variants finished in roughly 10^{-5} seconds for instances with crossing number 1.

500 unique requests for crossing number computations and most of them were successfully answered. We will see an immediate application of this tool in the next section.

4.2 Bounded Degree in Crossing Critical Graphs

In this section, which is based [Bok+19b], we want to consider the minimal obstructions for low crossing number. Unlike as, e.g., for genus, the class of graphs with prescribed crossing number at most $k \in \mathbb{N}$ is not closed under minor operations. As such, it does not follow that there is a finite set of forbidden minors for each k and a polynomial-time algorithm to test whether $cr(G) \leq k$ for any graph G. One way to algorithmically tackle the problem is to solve it by means of ILPs (cf. Section 4.1). However, the known formulations only consider Kuratowski subdivisions (i.e., 1-crossing-critical graphs, see below) to achieve dual bounds. Finding a subgraph that is c-crossing-critical, with $c > 1$, immediately implies a potentially stronger constraint that may be used in this approach. Clearly, it is also interesting from a purely theoretical standpoint to understand what exactly makes

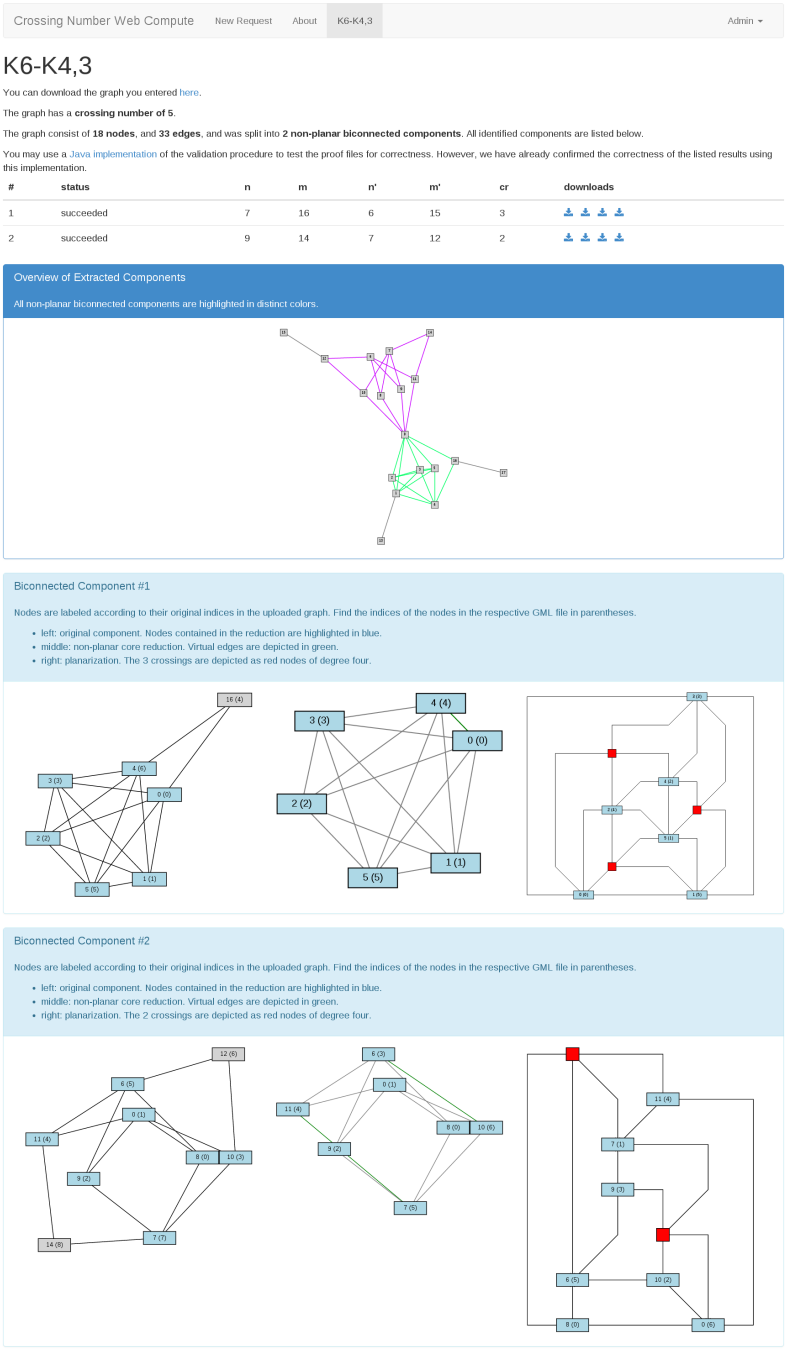


Figure 4.2: User Interface for Retrieving Proofs of Crossing Numbers

graphs have high crossing number. We thus consider the following definition:

Definition 4.2.1 (Crossing-criticality)**.** *A graph G is* c-crossing-critical *if its crossing number $cr(G)$ is at least c but removing any of its edges e decreases it below c, i.e., $cr(G \boxminus e) < c$ for each $e \in E(G)$. Conversely, in case of an (integrally) edge-weighted graph G, instead of the latter condition, we demand that decreasing the weight on any of G's edges by one also decreases its weighted crossing number. Similarly, for general graphs G, an edge is* critical *if its removal (unweighted) or a decrease of its weight also decreases the crossing number of G.*

Crossing-critical graphs are subject to mathematical studies since 1984 when Širáň initiated the field by constructing c-crossing-critical graphs for each value of c [Sir84]. Other researches build upon this result and broadened the known set of crossing-critical graphs, also showing nonexistence of such graphs with certain parameters [Bok+16; Bok+19a; Bok10; DHM18; Hli03; HST12; PR03; RT93]. In the setting of crossing critical graphs, one usually considers graphs without cut nodes as crossing number is additive over connected components. For similar reasons of preprocessing (cf. NPC, presented in [CG09]) a typical (and stronger than the above) restriction is to allow edge-weights while disregarding graphs that are not 3-connected. In 2003, consistent with all known crossing-critical graphs at that time, Richter conjectured the existence of a function $f : \mathbb{N} \to \mathbb{N}$ such that each node of a c-crossing-critical graph has degree at most $f(c)$, i.e., that the maximum node degree is somehow bounded in the criticality [MNW07]. This conjecture was later refuted by Dvořák and Mohar [DM10]. However, their proof is not constructive and as such no example of a high degree c-crossing-critical graph can be deduced from it.

We recently showed that indeed the conjectured function f does exist when restricted to small criticality. More precisely, it exists exactly for c at most 12 [Bok+19b]. In this section, we aim to—constructively—describe, for each c at least 13, a family of c-crossing-critical graphs that have arbitrarily high node degree.

4.2.1 13-Crossing-Critical Graphs with Large Degree

Let us define a family of graphs where each graph of this family is 13-crossing-critical and for each prescribed node degree $d \in \mathbb{N}$, there is a member that has node degree at least d.

Definition 4.2.2. *Assume an integer $k \geq 1$. Let C_u be a 6-cycle on the node set $\{x, u_1, u_2, u_3, u_4, u_5\}$ with edges xu_1, u_1u_2, u_2u_3, u_3u_4, u_4u_5, u_5x of*

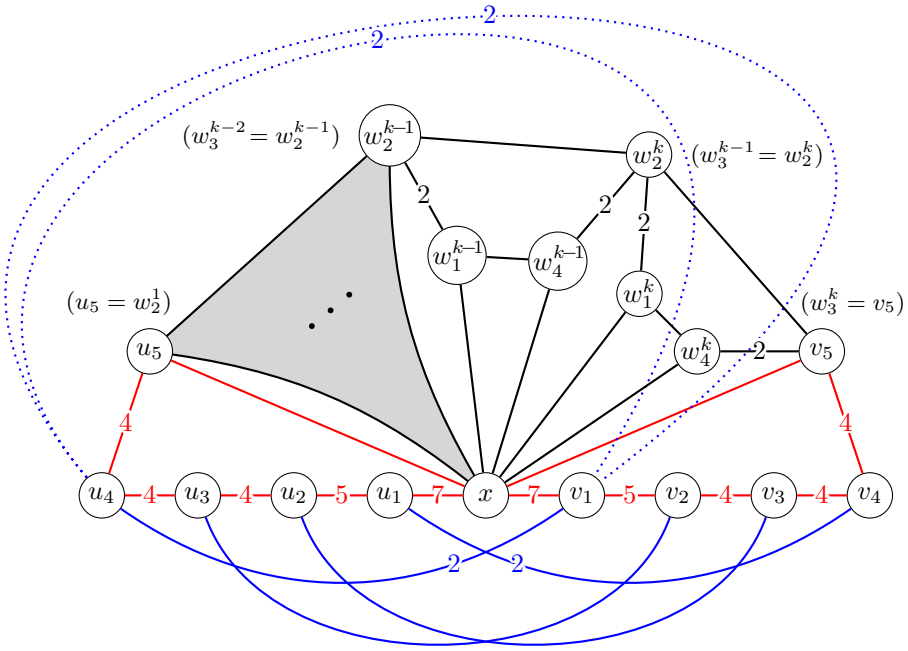


Figure 4.3: The graph G_{13}^k of Definition 4.2.2, drawn with 13 crossings. The weighted edges of this graph have their weight written as numeric labels, and all the unlabeled edges have weight 1. The bowtie part of this graph is drawn in red and blue (where blue edges are those between u_i and v_j nodes), and the wedges are drawn in black. Only the $(k-1)$-th and kth wedges are detailed, while the remaining wedges $1, ..., k-2$ analogously span the gray shaded area. Dotted lines show possible alternate routings of the edge v_1u_4 (each such routing preserves the optimal number of 13 crossings).

weight $7, 5, 4, 4, 4, 1$ *in this order. Analogously, let* C_v *be a* 6*-cycle on the node set* $\{x, v_1, v_2, v_3, v_4, v_5\}$ *isomorphic to* C_u *in this order of nodes (and with the same edge weight). The graph* B *is obtained by* $(C_u \sqcup C_v) \boxplus u_2v_3 \boxplus u_3v_2 \boxplus u_1v_4 \boxplus u_4v_1$ *where the former (latter) two new edges are assigned weight* 1 *(weight* 2*, respectively). Note that* $V(C_u) \cap V(C_v) = \{x\}$*, i.e., graph* B *has* 11 *nodes and the two cycles meet exactly at* x*.*

For $i \in [k]$*, let* D_i *denote the graph on the node set* $\{x, w_1^i, w_2^i, w_3^i, w_4^i\}$ *with the edges* xw_1^i*,* xw_4^i*,* $w_1^iw_4^i$*,* $w_2^iw_3^i$*, each of weight* 1 *and the edges* $w_1^iw_2^i$ *and* $w_3^iw_4^i$*, both of weight* 2*. From the union* $B \sqcup \big(\bigsqcup_{i\in[k]} D_i\big)$*, again identifying both nodes* x *in each operation, we obtain the graph* G_{13}^k *by identifying* u_5 *with* w_2^1 *and* w_3^k *with* v_5 *as well as* w_3^i *with* w_2^{i+1} *for each* $i \in [k-1]$*.*

Our construction is illustrated in Figure 4.3. We remark that $cr(G_{13}^1) \leq 12$, and for this reason we will assume $k \geq 2$ whenever referring to G_{13}^k. For future reference, we will call B the *bowtie* of G_{13}^k, and D_i the *ith wedge* of G_{13}^k.

Observation 4.2.3. *We have the following facts:*

1. *The graph* G_{13}^k *is* 3*-connected and non-planar.*
2. *The degree of node* x *in* G_{13}^k *is* $2k + 16$*.*
3. *There exists an automorphism of* G_{13}^k *exchanging* u_i *with* v_i *for* $i \in [5]$*.*

It remains to prove that each G_{13}^k is 13-crossing-critical. To this end, it suffices to show two claims; fist, that $cr(G_{13}^k) \geq 13$ holds, and second, that for every edge e of G_{13}^k, we have $cr(G_{13}^k \boxminus e) \leq 12$.

Lemma 4.2.4. *We have* $cr(G_{13}^2) = cr(G_{13}^3) = 13$*.*

Proof. Figure 4.3 shows a drawing of G_{13}^k with 13 crossings. For the—arguably more interesting—lower bounds, we use the aforementioned crossing number proof system, cf. Section 4.1. The respective computational results can be viewed at `http://crossings.uos.de/job/PZPmFDmDEKsgxLpftZmlXw` ($k = 2$) as well as `http://crossings.uos.de/job/EDsMIoyqrgonXEeD0plqdg` ($k = 3$). Since the proof system uses indices to label nodes, we provide a mapping from the proof files to our naming scheme: Node x is labeled 0. Cycle C_u uses nodes $0, 1, 2, 3, 4, 10$. Cycle C_v uses nodes $0, 5, 6, 7, 8$, and 14 for $k = 2$ (resp. 17 for $k = 3$). We remark that the proof file for $k = 3$ is particularly large, it spans 2150 cases with 717 Kuratowski constraints per case on average. For $k = 2$, there are just 1300 cases with about 180 constraints each. □

Now that the base cases are established, let us consider arbitrary values for k.

Lemma 4.2.5. *For every $k \geq 2$, it holds that $cr(G_{13}^k) = 13$.*

Proof. As for $k \in \{2, 3\}$, we still have $cr(G_{13}^k) \leq 13$ by Fig. 4.3. For the lower bound, we perform an induction on k, where the base cases, i.e., $k \in \{2, 3\}$, are proved in Lemma 4.2.4. Hence, we may assume that $k \geq 4$.

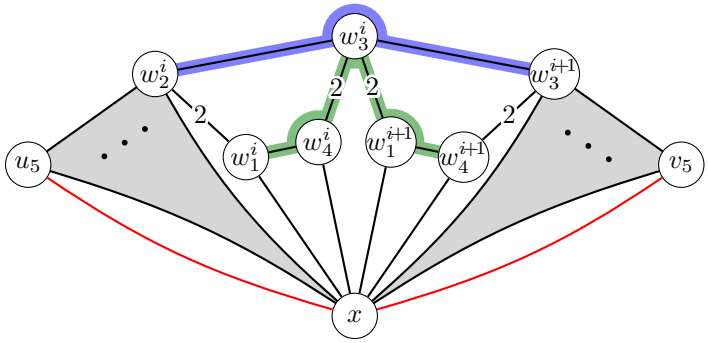
Consider a drawing of G_{13}^k with $c \coloneqq cr(G_{13}^k)$ crossings. Let $i \in [k-1]$, and recall that $w_3^i = w_2^{i+1}$. We now distinguish three cases based on the rotation around nodes w_3^i (the orientation is not important here and we consider any sequence of incident edges equal to its inverse). To this end, recall that no pair of adjacent edges crosses in an crossing-minimal drawing. As such, we may sensibly speak about the rotation systems of drawings that involve crossings (although they are not planar). When referring to the cyclic order around a node v, one may also think about this as restricting the drawing to a sufficiently small area around v that contains no crossings.

1. There exists an $i \in [k-1]$, such that the edges incident to $w_3^i = w_2^{i+1}$, have the cyclic order $w_3^i w_4^i$, $w_3^i w_1^{i+1}$, $w_3^i w_3^{i+1}$, $w_3^i w_2^i$. See Fig. 4.4a, where this cyclic order is anti-clockwise. In this case, we draw a new edge $w_1^i w_4^{i+1}$ along the path $(w_1^i, w_4^i, w_3^i, w_1^{i+1}, w_4^{i+1})$, and another new edge $w_2^i w_3^{i+1}$ along the path $(w_2^i, w_3^i, w_3^{i+1})$ and assign weight 1 to both new edges. Then, we remove the nodes w_4^i, w_3^i, w_1^{i+1} and their incident edges from the drawing. The new drawing depicts a graph that is isomorphic to G_{13}^{k-1}—the ith and $(i+1)$-th wedge have been replaced by just one new wedge.

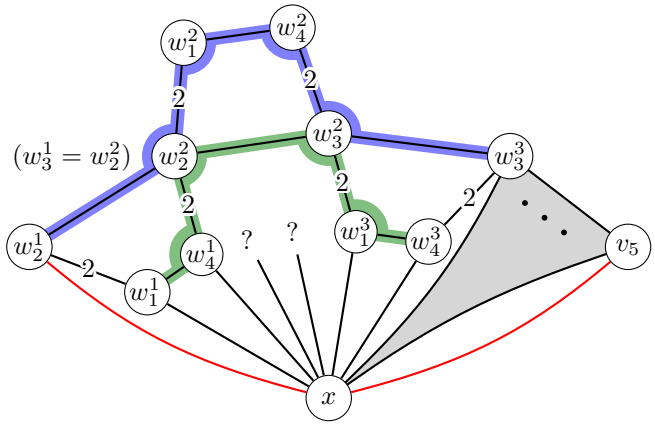
   Moreover, thanks to our assumption, we avoid crossings between $w_1^i w_4^{i+1}$ and $w_2^i w_3^{i+1}$ in the considered area around the former w_3^i. Therefore, every crossing of the new drawing (including possible crossings of each of the new edges $w_1^i w_4^{i+1}$ and $w_2^i w_3^{i+1}$ among themselves or with other edges) existed already in the original drawing of G_{13}^k, and hence $cr(G_{13}^{k-1}) \leq c$. However, $cr(G_{13}^{k-1}) \geq 13$ by the induction assumption, and it follows that $c \geq 13$ holds true in this case.

2. The same proof as above works if the cyclic order around w_3^i is $w_3^i w_4^i$, $w_3^i w_1^{i+1}$, $w_3^i w_2^i$, $w_3^i w_3^{i+1}$.

3. Otherwise, for each $i \in [k-1]$, the edges incident to $w_3^i = w_2^{i+1}$ have the cyclic order $w_3^i w_4^i$, $w_3^i w_3^{i+1}$, $w_3^i w_1^{i+1}$, $w_3^i w_2^i$. See Fig. 4.4b. We will use this assumption only for $i \in [2]$ as follows.

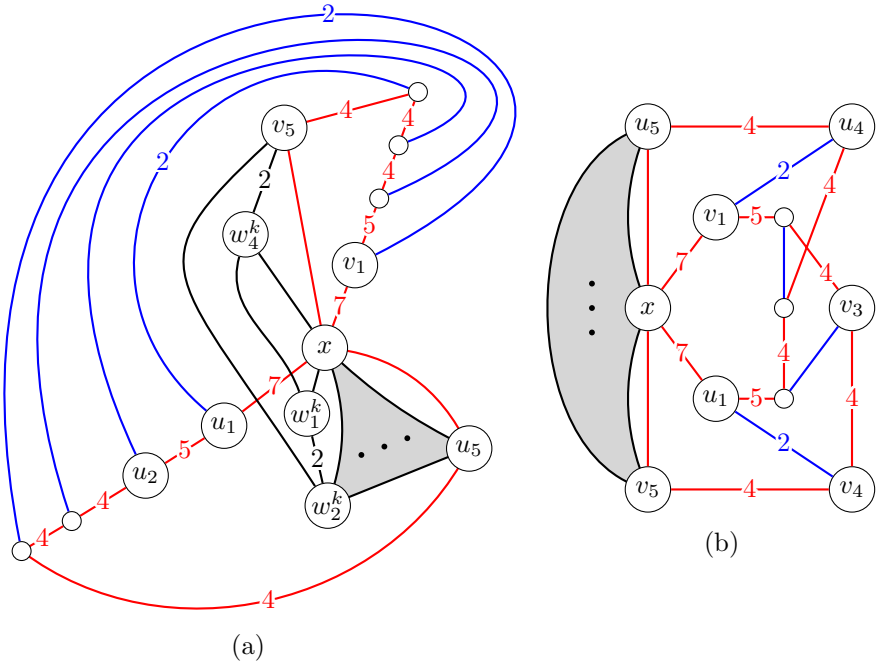


(a) We "shrink" two wedges into one by drawing new edges $w_1^i w_4^{i+1}$ (green) and $w_2^i w_3^{i+1}$ (blue) along the depicted paths.



(b) We likewise "shrink" three wedges into one by drawing the depicted new edges $w_1^1 w_4^3$ and $w_2^1 w_3^3$ (this picture does not specify how w_1^2 and w_4^2 connect to x since it is not important for us).

Figure 4.4: Two cases of the induction step in the proof of Lemma 4.2.5.



(a) (b)

Figure 4.5: Two drawings of the graph G_{13}^k, having (a) 14 and (b) 16 crossings. These drawings are used to argue criticality of some of the bowtie (red) edges of G_{13}^k. The gray areas span the crossing-free wedges of G_{13}^k which are not detailed in the pictures, similarly as in Fig. 4.3.

We draw a new edge $w_1^1w_4^3$ along the path $(w_1^1, w_4^1, w_3^1, w_3^2, w_1^3, w_4^3)$, and another new edge $w_2^1w_3^3$ along the path $(w_2^1, w_3^1, w_1^2, w_4^2, w_3^2, w_3^3)$ and assign weight 1 to both new edges. Then, we remove the nodes $w_4^1, w_3^1, w_1^2, w_4^2, w_3^2, w_1^3$ and their incident edges from the drawing. The new drawing represents a graph which is now isomorphic to G_{13}^{k-2}—the first, second, and third wedge have been replaced with just one new wedge.

As in the previous case, we can avoid crossings between $w_1^1w_4^3$ and $w_2^1w_3^3$ in the considered area around the former w_3^1 and w_3^2. Therefore, analogously, $cr(G_{13}^{k-2}) \leq c$. However, $k-2 \geq 2$ and $cr(G_{13}^{k-2}) \geq 13$ by assumption, and hence $c \geq 13$ holds true also in this case.

This concludes our induction on k. □

Lemma 4.2.6. *For each number $k \in \mathbb{N}$ and edge $e \in E(G_{13}^k)$, we have $cr(G_{13}^k \boxminus e) \leq 12$.*

Proof. On a high level, our proof strategy is to provide a collection of drawings of G_{13}^k such that for each edge of G_{13}^k there is a drawing, where the removal of that edge yields a drawing with less than 13 crossings. Note that for edges with weight more than one, we will decrease its weight by one instead of removing it entirely.

We observe that already Fig. 4.3, which we used to illustrate the construction of G_{13}^k, proves the claim for $e \in \{u_1v_4, u_2v_3, u_3v_2, u_4v_1\}$ (the blue edges in Fig. 4.3); whenever any single one of these edges is removed/decreased, we save at least one crossing. In fact, Fig. 4.3 also proves the claim for $e \in \{xv_5, v_4v_5\}$ (see the dotted routings of the edge v_1u_4 in Fig. 4.3), and hence also for $e \in \{xu_5, u_4u_5\}$ by symmetry (cf. Observation 4.2.3).

To proceed with the remaining edges of the bowtie graph (the red edges in Fig. 4.3), we resort to non-optimal drawings, i.e., drawings that have more than 13 crossings. However, for each relevant edge e, these non-optimal drawings will still provide a drawing of $G_{13}^k \boxminus e$ with at most 12 crossings. Let us first consider the drawing obtained from the one of Fig. 4.3 by "flipping" the right-hand part of the picture along the line through x and u_5, see Fig. 4.5a. In this drawing (with 14 crossings), the only crossings occur between edge xu_1 (weight 7) and the edges $w_1^kw_4^k$, $w_2^kv_5$. By a slight shift of the latter two edges, we obtain another drawing with only 14 crossing between the edges u_1u_2 (weight 5), u_1v_4 (weight 2) and again the edges $w_1^kw_4^k$, $w_2^kv_5$. Decreasing the weight of xu_1 (respectively, of u_1u_2) drops the crossing number down to 12. Second, there is a drawing with 16 crossing that occur only between the edges v_2v_3 and u_3u_4 (both of weight 4); see Fig. 4.5b. Decreasing v_2v_3 or u_3u_4 again drops the crossing number down to at most 12. Consequently, our claim holds for $e \in \{xu_1, u_1u_2, v_2v_3, u_3u_4\}$ and—by symmetry—for $e \in \{xv_1, v_1v_2, u_2u_3, v_3v_4\}$.

We are left with the last and perhaps most interesting cases where the considered edge e is an edge in the ith wedge D_i. Imagine we "disconnect" D_i by temporarily removing nodes w_1^i and w_4^i and the edge $w_2^iw_3^i$, and then twist the bowtie graph B together with the adjacent strips of wedges $\bigsqcup_{j\in[i-1]} D_j$ and $\bigsqcup_{j\in[k-i]} D_{i+j}$, removing all crossings. Upon reintroducing the nodes and edges of D_i, we may obtain the drawing of Fig. 4.6a with 13 crossings that are only between the edges xw_1^i, $w_2^iw_3^i$, $w_1^iw_4^i$ and the six blue bowtie edges. For each edge $e \in \{xw_1^i, w_2^iw_3^i, w_1^iw_4^i\}$, its removal from G_{13}^k thus decreases the crossing number to at most 12. Lastly, we may move node w_1^i to obtain another drawing, see Fig. 4.6b. The latter drawing has 18 crossings between the edges $w_1^iw_2^i$, $w_2^iw_3^i$ and the six blue bowtie edges. Again, removing one edge of $w_1^iw_2^i$ thus drops the crossing number down to 12. By symmetry, the claim is thus proved also for each $i \in [k]$ and edge $e \in E(D_i)$. □

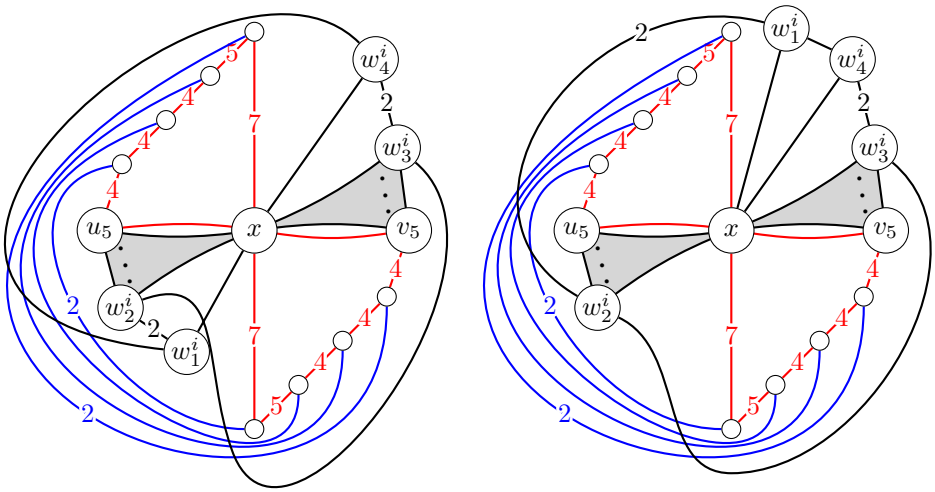


Figure 4.6: Two drawings of the graph G_{13}^k, having 13 (a) and 18 (b) crossings. These drawings are used to argue criticality of edges of the ith wedge. The gray areas span the crossing-free wedges of G_{13}^k which are not detailed in the pictures, similarly as in Fig. 4.5.

From the above considerations on G_{13}^k, we are now able to conclude the following.

Theorem 4.2.7. *For every positive integer $d \in \mathbb{N}$, there exists a 3-connected (edge-weighted) 13-crossing-critical graph, with maximum degree at least d.*

4.2.2 Arbitrary Criticality and Many Large-Degree Nodes

In the previous section, we have constructed an infinite family G_{13}^k of 13-crossing-critical graphs with unbounded maximum degree. Next, we want to study the following two natural questions.

(a) Do similar c-crossing-critical families exist for each $c > 13$? (Recall that the maximum node degree in c-crossing-critical graphs is upper bounded for $c < 13$.)

(b) Are there c-crossing-critical graphs with more than one node of high degree? (In G_{13}^k, only node x has arbitrarily high degree.)

Let us address these questions with some extended constructions based on the previously presented ideas. Clearly, regarding question (a), for each $c, d \in \mathbb{N}$

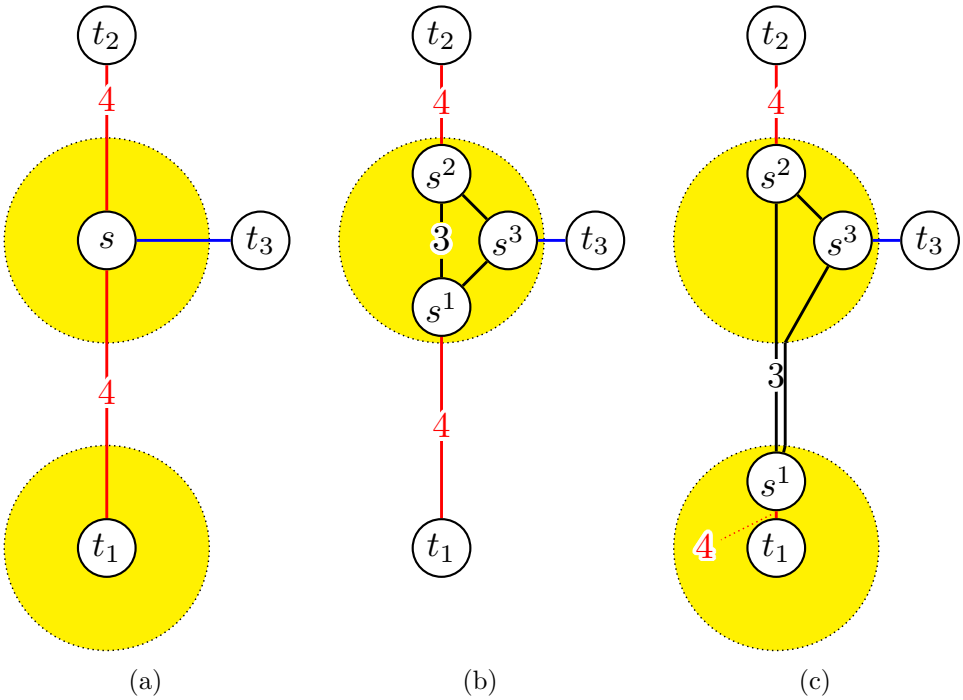


Figure 4.7: Before (a) and after (b) a 4-to-3 expansion is applied at node s. The yellow circle represents a crossing-free area that we may assume around each node. Figure (c) depicts the same graph as (b) but shows a different drawing.

with $c \geq 14$, we may obtain a c-crossing-critical graph with maximum degree greater than d simply by constructing the union of G_{13}^k with $(c-13)$-many disjoint copies of $K_{3,3}$. Similarly, regarding question (b), we may consider the union of t disjoint copies of G_{13}^k to obtain a $13t$-crossing-critical graph with t nodes of arbitrarily high degree. However, we are interested in 3-connected graphs, i.e., in those graphs that cannot trivially be partitioned into smaller such graphs, and both of the above constructions result in disconnected ones.

Definition 4.2.8. *Let G be an edge-weighted graph and $s \in V(G)$ a node incident exactly with two edges st_1 and st_2 of weight 4, and one edge st_3 of weight 1. The following operation is called a* 4-to-3 expansion *at s: Remove node s with its incident edges from G, and add three new nodes s^1, s^2, s^3, two new edges t_1s^1, t_2s^2, each of weight 4; one new edge s^1s^2 of weight 3, and three edges t_3s^3, s^1s^3, s^2s^3, each of weight 1. See Fig. 4.7 for an illustration of this operation.*

We define H^k_{13} as the graph that is constructed from G^k_{13} by applying a 4-to-3 expansion to each of the two nodes v_3 and u_3. (cf. Figure 4.3).

Construction H^k_{13} will allow us to use the well-established operation *zip product* to join multiple such graphs and hence to obtain many nodes of high degree while still maintaining 3-connectivity and crossing-criticality.

Lemma 4.2.9. *Consider a 13-crossing-critical graph G and a node $s \in V(G)$ of degree 3 that has two incident edges of weight 4 and one of weight 1. The graph G' that is obtained by a 4-to-3 expansion of G at s is also 13-crossing-critical.*

Proof. First, we will show that for each edge $e \in E(G')$ we have $cr(G' \boxminus e) \leq 12$, i.e., the crossing number of G' drops below 13 when decreasing the weight of edge e by one. If also $e \in E(G)$, i.e., if e is an original edge of G, then we consider a drawing $\mathcal{D}$ of $G \boxminus e$ with at most 12 crossings. Such a drawing $\mathcal{D}$ exists since G is 13-crossing-critical.

Clearly, we may perform the 4-to-3 expansion in a small crossing-free area around s in $\mathcal{D}$ without introducing any additional crossings (cf. Figs. 4.7a and 4.7b). Hence, it only remains to consider each edge e that is incident to a node in $S \coloneqq \{s^1, s^2, s^3\}$.

For edges that have exactly one incident node in S, i.e., edges $t_i s^i$ ($i \in [3]$), we proceed as for the previous case, but start with a drawing of $G \boxminus t_i s$.

Finally, for edges that have both endpoints in S, say $s^1 s^2$ or $s^1 s^3$, we once again perform a 4-to-3 expansion, this time starting with a drawing of $G \boxminus t_1 s$. By placing s_1 close to t_1 and routing $s^1 s^2$ and $s^1 s^3$ in parallel along the previous path of $t_1 s$, we again obtain a drawing of $G' \boxminus e$ for each $e \in \{s^1 s^2, s^1 s^3\}$ (cf. Figs. 4.7a and 4.7c). The case $e = s^2 s^3$ is symmetric (using original $t_2 s$).

Second, we show that $c = cr(G') \geq 13$. We consider an optimal drawing $\mathcal{D}$ of G' with c crossings. By a folklore argument, we may assume the new triangle $T \coloneqq G'[S]$ to be drawn without crossings.

We define a as the sum of weights of crossing edges over each edge in T. Similarly, for each $i \in [2]$, let b_i denote the sum of weights of edges crossing $s^i s^3$.

If $b_1 + b_2 \leq a$, we modify $\mathcal{D}$ by rerouting $s^1 s^2$ along the path $\langle s^1, s^3, s^2 \rangle$. We may draw along the left- or right-hand side of this path. From this new drawing $\mathcal{D}_1$ we derive a drawing of G with c crossings by placing s at the position of s_3 and routing $t_1 s$ along the path $\langle t_1, s^1, s^3 \rangle$, $t_2 s$ along $\langle t_2, s^2, s^3 \rangle$, and $t_3 s$ along $t_3 s_3$. Since G has crossing number 13, we have $c \geq 13$ or $b_1 + b_2 > a$.

Consequently, we assume $b_1 + b_2 \geq a + 1$. If $b_2 \geq a$ (and $b_1 \geq a$ is a symmetric case), we modify $\mathcal{D}$ by rerouting the path $\langle s^1, s^3, s^2 \rangle$ along the edge $s^1 s^2$, and prolong the edge $s^3 t_3$ along $s^3 s^1$. This results in a drawing $\mathcal{D}_2$ with at most $c - (b_1 + b_2) + a + b_1 \leq c - b_2 + a \leq c$ crossings. Then, as in the previous, we turn $\mathcal{D}_2$ into a drawing of G with (again) at most c crossings, and so $c \geq 13$ holds, too. It remains to consider the case that $b_1, b_2 \leq a - 1$. Together with $b_1 + b_2 \geq a + 1$ we get that $a \geq 3$, and hence the number of crossings in $\mathcal{D}$ is at least $3a + b_1 + b_2 \geq 3a + a + 1 \geq 13$, as desired. □

We observe that the number 13 of crossings in Lemma 4.2.9 is rather specific and the claim cannot be easily generalized to other numbers of crossings. For instance, one can construct a graph of crossing number 14, such that one of its 4-to-3 expansions has crossing number only 13.

Corollary 4.2.10. *For every $k \geq 2$, the graph H_{13}^k is 13-crossing-critical.*

By the above considerations and the additivity of c-crossing-criticality over zip products, we are now in a position to answer our questions about higher crossing-criticality and the number of nodes with high degree.

Corollary 4.2.11. *For every two integers $c \geq 13$ and $d \geq 1$, there exists a 3-connected c-crossing-critical graph, whose maximum degree is at least d.*

Proof of Corollary 4.2.11. We construct a family of such graphs where we denote the respective members by $G(c, d)$. Let $G(13, d) \coloneqq H_{13}^{\lfloor d/2 \rfloor}$. Note that $G(13, d)$ contains two nodes of degree 3. For $c > 13$, we proceed by induction, assuming that we have already constructed the graph $G(c - 1, d)$ and it contains a node of degree 3. We construct $G(c, d)$ as a zip product of $G(c - 1, d)$ and the $K_{3,3}$. It is c-crossing-critical and it still contains the same node x of high degree as H_{13}^k does. Furthermore, $G(c, d)$ contains a new node of degree 3 from the $K_{3,3}$. □

Corollary 4.2.12. *For any integers $c \geq 13$ and $i \in [\lfloor c/13 \rfloor]$, there exists a 3-connected c-crossing-critical graph that has i nodes of degree at least d.*

Proof of Corollary 4.2.12. We denote the claimed graph by $G(c, d, i)$. The proof proceeds in a manner similar to the proof of Corollary 4.2.11. This time we inductively zip together i copies of the graph $H_{13}^{\lfloor d/2 \rfloor}$ and $(c - 13i)$-many copies of the $K_{3,3}$. This results in a c-crossing-critical graph with i nodes (one per each $H_{13}^{\lfloor d/2 \rfloor}$-copy) of degree greater than d. Note that we never "run out" of degree-3 nodes in the construction since each copy of $H_{13}^{\lfloor d/2 \rfloor}$ has two such nodes. □

Conclusion. We have seen that there are indeed graphs with high crossing criticality and arbitrarily high node degree: the set of minimal obstructions for low crossing number is even more diverse than previously expected. One may see this as another hint for the intricacy of the crossing number problem and as a further obstacle when trying to quickly find large crossing-critical subgraphs to facilitate the computation of crossing numbers.

4.3 Extending Simple Drawings

In this section that is based on [Arr+20], we study the problem of inserting a new edge into a drawing.

More precisely, consider a simple drawing $\mathcal{D}'$ of a (simple, undirected) graph G and the drawing $\mathcal{D}$ that is obtained by removing an arbitrary edge $e \in E(G)$ from $\mathcal{D}'$. We say that $\mathcal{D}'$ is a *simple extension* of $\mathcal{D}$. Recall that a drawing is simple if adjacent edges do not cross each other and each pair of edges crosses at most once (cf. Section 1.4). We want to study the following decision problem that is denoted simple single edge insertion (SSEI): Given a simple drawing $\mathcal{D}$ of a graph G and a pair of non-adjacent nodes $u, v \in V(G)$, does there exist a simple extension of $\mathcal{D}$ that contains the edge uv?

A similar but more general problem is studied in [ADP19]. There, the authors consider an arbitrarily large set of edges, each of which has to be inserted while maintaining simplicity of the overall drawing. They show that (1) the problem is NP-hard (assuming the number of edges to be part of the input), and (2) the problem can be solved in polynomial time if we ask for the insertion of a single edge uv and $\{u, v\}$ is a dominating set in the input graph.

We strengthen the former result in the sense that the problem remains NP-hard even for the insertion of a *single* (general) edge. We will provide a reduction from 3-SAT to SSEI.

Transformed 3-SAT. Since our main reduction neither works with clauses that contain only positive literals nor with those that contain only negative ones, we need a small reduction to a transformed 3-SAT (T3-SAT) problem first:

Observation 4.3.1. *The following transformation of a clause with only positive (negative) literals, preserves the satisfiability of the clause (symbol y*

denotes a new variable and $\bot$ is the constant truth value `false`*):*

$$x_i \vee x_j \vee x_k \Rightarrow \begin{cases} x_k \vee y \vee \bot & (i) \\ x_i \vee x_j \vee \neg y & (ii) \end{cases} \quad \neg x_i \vee \neg x_j \vee \neg x_k \Rightarrow \begin{cases} \neg x_i \vee \neg x_j \vee y & (iii) \\ \neg x_k \vee \neg y \vee \bot & (iv) \end{cases}$$

4.3.1 Reduction from T3-SAT to SSEI.

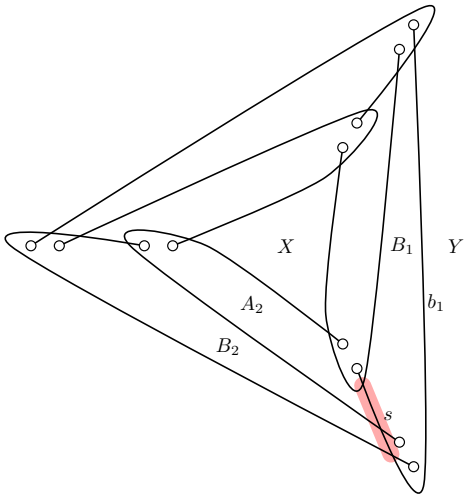
To provide some intuition on the reduction from T3-SAT, we summarize our proof strategy as follows: We construct the drawing of G such that there exists a small horizontal region, called the *corridor* (cf. Fig. 4.8b) that the new edge uv has to be inserted into. Starting on the left side of the corridor, uv must first pass through a set of *variable gadgets*, each corresponding to a variable of the given 3-SAT instance $\mathcal{F}$. For each such gadget, an assignment of the respective variable can be derived by inspecting the local routing of uv through the gadget. The remaining corridor contains a *clause gadget* for each clause of $\mathcal{F}$. For each clause gadget, uv can pass through the gadget if and only if the respective clause is satisfied by the variable assignment induced by the routing through the variable gadgets. Essentially, a crossed edge in a variable gadget will correspond to a false literal and a crossed edge in a clause gadget to a satisfying, i.e., true, literal.

Let us begin by describing how to obtain the claimed corridor.

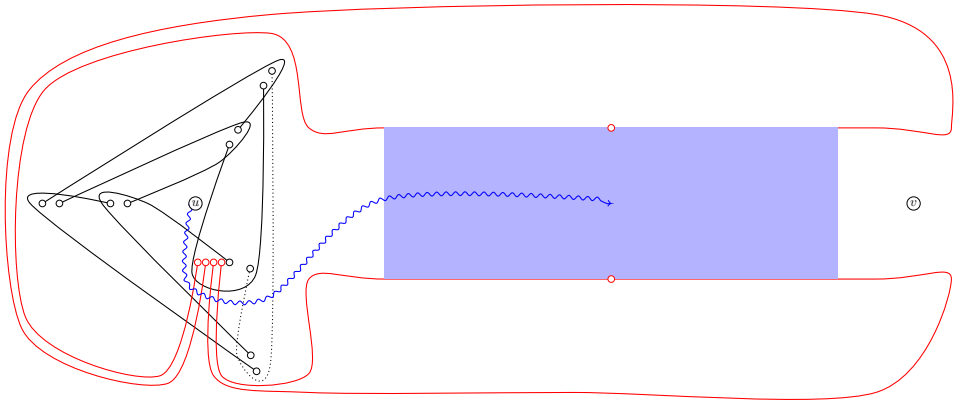
Construction of Corridor. In [Kyn+15, Fig. 11], a simple drawing of a matching that consists of just 6 edges is presented. We denote this particular drawing by ◎ and repeat it (in a slightly homoemorphically transformed manner) as Fig. 4.8a for the reader's convenience. To obtain the base drawing ◎′ that defines our corridor, cf. Fig. 4.8b, we place two new nodes u, v in the former regions X, Y, respectively; remove the edge b_1; and introduce 4 new bounding edges that each have to be crossed by uv close to u to reach v.

By the construction of Kynčl et al. it is clear that any edge passing from X to B_1 in ◎ has to cross the segment s in Fig. 4.8a since otherwise it would be possible to route a curve from X to Y. We use this fact to place the 4 new edges that each must be crossed once. They are used to bound the corridor.

Next, we want to describe our gadgets. Each such gadget resides in a small vertical strip within the corridor, occupying its full height and without overlapping with any other gadgets. Later (see paragraph "assembly"), we will connect our gadgets using the two regions directly above and below the corridor (each essentially bounded by two red edges).



(a) The drawing ◎. There is no curve from X to Y in a simple extension of ◎. Any curve that enters region B_1 from region X must cross segment s (shaded red).



(b) The drawing ◎′ that is derived from ◎. Edge b_1 (dotted) is removed. The 4 new (red) edges are introduced at the former segment s. Any routing of edge uv (indicated as blue wiggly curve) in a simple extension of ◎′ passes through the (shaded blue) corridor since it cannot cross any of the red edges twice.

Figure 4.8: Construction of the corridor (b) from a known drawing (a).

Constant False Gadgets. The two *constant false gadgets*, cf. Fig. 4.9a, each consist of a bundle of edges that each have to be crossed inside the false gadget and hence cannot be crossed later on in the respective clause gadget. For the shown orientation, integer k denotes the total number of constant false literals occurring in clauses of type (iv) $\mathcal{F}$. For the clauses of type (i) the gadget is vertically flipped and k again denotes the respective number of occurrences.

Variable Gadgets. For each variable in $\mathcal{F}$, there is a *variable gadget*, cf. Fig. 4.9b. It consists of a single center node and two bundles of edges incident with it and crossing over either border such that the top (bottom) bundle consists of j (ℓ, respectively) edges. Here, j (ℓ) denotes the number of positive (negative, respectively) literals of this variable over all clauses.

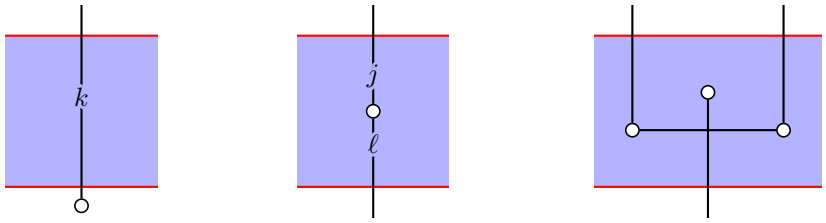
We note that uv, when passing through a variable gadget, either crosses all its edges corresponding to positive literals or all its edges corresponding to negative literals (and either option is locally feasible).

Clause Gadgets. For each clause in $\mathcal{F}$, there is a *clause gadget*, cf. Fig. 4.9c. It consists of four edges, three of which each cross the boundary of the corridor such that each crossing edge corresponds to exactly one literal occurrence in the clause. Positive literals cross over the top border while negative ones cross the bottom one. The fourth (horizontal) edge runs between the nodes incident with the unique two edges that share a common (top or bottom) crossed border. The only crossing that appears within the gadget is between the horizontal edge and the single edge crossing the opposite border.

We observe that uv, when passing through a clause gadget, has to cross at least one edge corresponding to a literal (and for each literal it is locally feasible to cross only this literal).

Assembly. To construct our instance from $\mathcal{F}$ we start with ◎′ and along the corridor from left to right we first place the two constant false gadgets, one in each orientation. Next, we place a gadget for each variable $x_1, x_2, \cdots, x_n$. After placing all variable gadgets, we move along the corridor until we pass the two red nodes on the corridor's border. This is crucial as otherwise the edges connecting variables (and constant false) gadgets with clause gadgets would not be able to cross the red border twice.

Finally, we place all clause gadgets (in arbitrary order along the corridor). Fig. 4.10 schematically explains how the variable (and constant false) gadgets are connected to the clause gadgets: Each non-constant literal occurrence



(a) Const. gadget. (b) Variable gadget. (c) Clause gadget.

Figure 4.9: Different gadgets used in our construction. Labeled lines denote bundles of edges with the respective cardinality. We will establish the routing of the edges crossing to the outside of the corridor later.

has a corresponding edge in a clause gadget and one in a variable gadget. We identify these edges for each literal occurrence without modifying their routings in the corridor. By placing each literal's edges on a unique horizontal line in parallel to the corridor we easily achieve a simple drawing where no edge has to wrap around the corridor. The only exception from this routing is $\perp$ whose literals occupy two horizontal lines, one above and one below the corridor.

By the above constructions and conclusions we are now able to state this section's primary result:

Theorem 4.3.2. *SSEI is* NP*-hard.*

Corollary 4.3.3. *SSEI remains* NP*-hard even when restricted to simple drawings of* matchings*.*

Proof. Consider any node w with its incident edges in a general simple drawing $\mathcal{D}$ where we want to insert a new edge uv. Note that we may assume a non-empty crossing-free (circular) region $\bigcirc$ around w. We will modify $\mathcal{D}$ to obtain $\mathcal{D}'$ such that its relevant properties are retained but instead of w, there will be $\deg(w)$-many leaves: This is readily obtained by (1) placing $\deg(w)$-many new nodes in $\bigcirc$, each corresponding to an edge incident with w; (2) for each such node detaching its corresponding edge from w and attaching it to the new node instead, meanwhile maintaining the same routing outside of $\bigcirc$; and finally (3) modifying the drawing in $\bigcirc$ such that each pair of newly introduced edges crosses. We provide an example of this operation in Fig. 4.11. After applying it, all routings of uv remain the same as in $\mathcal{D}$, except for routings that pass through $\bigcirc$ in $\mathcal{D}'$. However, for each routing r through $\bigcirc$ in $\mathcal{D}'$, there is an equivalent routing along

(segments of) the border of $\bigcirc$ that may save some crossings but requires no additional crossings over r. $\square$

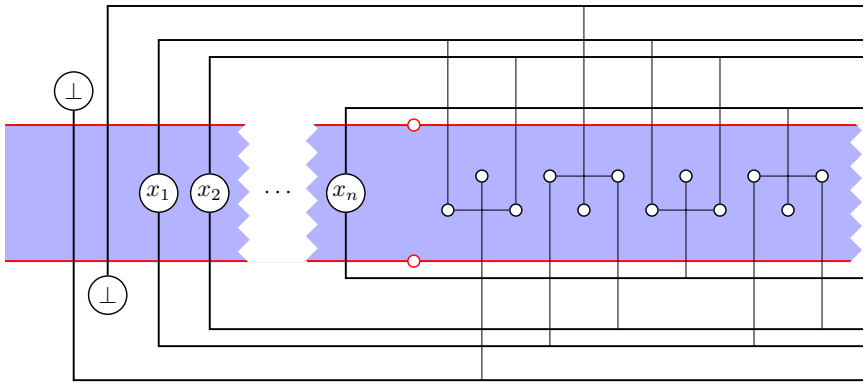


Figure 4.10: Placing the gadgets in the corridor and connecting them. The nodes of the two constant false gadgets are labeled $\bot$. Each variable gadget's node is labeled with the respective variable name. Thick black lines denote bundles of edges while regular ones are edges. Here, we exemplarily exhibit four clauses (types) in this order from left to right: $(x_1 \vee x_2 \vee \bot)$, $(\neg x_1 \vee \neg x_2 \vee \bot)$, $(x_1 \vee x_2 \vee \neg x_n)$, and $(\neg x_1 \vee \neg x_2 \vee x_n)$. It is easily seen that this construction maintains the drawing's simplicity.

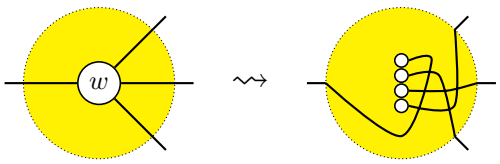


Figure 4.11: Transformation to a matching. The yellow region denotes the crossing-free region ◯. Note how the vertical order of nodes corresponds to the anti-clockwise order of edges incident with w.

Chapter 5

Conclusion and Outlook

Cycle (*baseball*). A single, a double, a triple, and a home run hit by the same player in the same game.

A primary objective of this thesis was to develop better algorithmic methods for the efficient computation of measures of non-planarity. We have achieved this goal in the sense that novel faster exact algorithms for maximum planar subgraph and graph genus were discovered. The mathematical programming models that these algorithms are based upon are provably stronger than the state-of-the-art and both algorithms perform orders of magnitude faster than the previously fastest algorithms.

For MPS, we discovered a particularly rich set of supplemental variables and constraints, based on small cycles, that are beneficial in both theory and practice. While there are viable alternatives derived from different planarity criteria, the known model based on Kuratowski subdivisions that was first described by Mutzel remains the foundation of the strongest choice in practice. We showed that—although there are good heuristics—there still is a need for even better heuristics and faster exact algorithms which is particularly evident when employing the planarization approach. Regarding the approximation of MPS, we have systematically identified a large set of outstanding obstacles to several rather general algorithmic approaches that each seem highly promising without thorough consideration.

In terms of genus, we discovered a stronger model that uses not just short cycles—as opposed to the strongest MPS model—but short closed walks instead. It is relatively similar to our strongest MPS model but requires some subtle differences: Most prominently, we could not base it on a completely

different model that is already sufficient in itself. Unfortunately, no further set of optional constraints that reliably improves practical performance could be identified.

In the realm of crossing numbers, we have revisited a proof extraction and verification procedure to make computations of crossing numbers transparent and comprehensible also for pure graph theorists. In addition, we showed that the set of crossing-critical graphs is surprisingly diverse. It contains graphs with arbitrarily high node degree that consists of specific planar subgraphs joined at a center node, its core building blocks are wedges. Finally, we showed that it is NP-hard to decide whether an edge can be inserted into a simple drawing while maintaining simplicity of the resulting drawing and without modifying the given drawing.

Naturally, our discoveries also present us with a new set of questions.

Approximation of MPS. We have seen a diverse set of obstacles that seem to hinder us from obtaining better approximations for maximum planar subgraph. Still, there is no hard evidence that an approximation with a ratio strictly better than 4/9 is not obtainable. Indeed, in the realm of practical algorithms (in this case: algorithms that do not require matroid subroutines), a recent result improved the long-standing ratio of 7/18 to 13/33 [CS17]. We wonder whether this factor can be further improved, possibly by iteratively considering even larger (outer-)planar subgraphs.

Approximation of Skewness. While it is known that skewness cannot be approximated within an arbitrarily good constant factor, no approximation algorithm has been discovered so far. Iteratively searching for Kuratowski subdivisions and removing them entirely leads to a factor that is bounded by the size of the Kuratowski subdivisions. Clearly, a difficulty arises when there are Kuratowski subdivisions of arbitrary size that cannot be eliminated by efficiently finding and removing smaller bounded ones. We raise the question whether such an approach could be used to obtain a constant factor approximation of skewness.

Separation for L.-R. Coloring Model. Among the MPS models that are not equipped with cycle-based extension, the practical performance of the left-right edge coloring model, based on the planarity criterion by de Fraysseix and Rosenstiehl, is second only to that of the model by Mutzel (using Kuratowski's characterization of planarity). However, as we already noted in Section 2.4, a contributing factor to the better performance of the

Mutzel model may be the far better separation procedure to identify many violated Kuratowski constraints in linear time. We ask whether the left-right planarity test can be modified such that a sensible rounding scheme with the modified test yields a linear-time separation procedure that identifies violated coloring restrictions with high probability.

Combination of Planarity Criteria. We have seen that the cycle model in combination with the Mutzel (MPS) and facial-walk (genus) model is surprisingly strong. For MPS, we studied models based on different planarity criteria, where each model is based on exactly one criterion. One may regard the cycle model as an amalgamation of two planarity criteria where we took a sufficient model for the problem at hand and lifted it to include (parts of) Whitney's planarity criterion, i.e. the existence of a corresponding dual. In a similar fashion we may combine (parts of) many models for MPS and genus. Can we find even faster models that focus on the most profitable elements from several criteria?

Fine-Grained Examination of Cycle Space. Another direction for improving the already very efficient cycle-based models is to consider not just a single maximum cycle length D but to use different values, each tied to a different subgraph (or even edge) of the input. We hope that such an approach can amplify the impact of the cycle model when the graph's density is highly non-homogeneous: Denser subgraphs in the input will likely result in subgraphs with short faces also in a maximum planar (overall) subgraph. Sparse subgraphs, however, likely require the enumeration of larger face candidates to solve the cycle model efficiently. It is not immediately clear how such a model would be designed but it certainly seems to be a worthwhile research direction.

Facets of New Models. Although we have identified—particularly for MPS—a large set of supplemental constraints that strengthen the existing models, it remains unclear if and which of these new constraints form facets of their respective polytopes. Hence, an even more thorough theoretical examination of our constraints may be in order. In addition, it would be interesting to see if and which constraints are dominated by others. For example, we know the Kuratowski-cycle constraints to strengthen the base cycle model for MPS. However, it is unclear whether they also strengthen the cycle model *with the pseudo-tree* extension (although likely).

Genus Algorithm by Brinkmann. A surprisingly fast and purely combinatorial algorithm for the genus problem was presented recently by Gunnar Brinkmann [Bri20]. It is based on a B&B scheme that iteratively inserts edges while carefully bounding the genus w.r.t. this operation. We wonder whether our models could be employed to cut off large potions of the resulting B&B tree. Since his branching procedure is incredibly fast, we cannot simply perform an LP-computation at each B&B-node. Instead, a sensibly accurate and similarly fast heuristic, to decide whether advanced bounding should be used, would have to be designed.

Cycle Model for Crossing Numbers. While we undertook some initial attempts to adapt the cycle model for the computation of crossing numbers (that we did not report here), it seems surprisingly hard to exploit the cycle structure to obtain better models for crossing numbers. Still, there are results that hint towards a similar improvement for crossing numbers. For example, the well-known crossing lemma that asymptotically bounds the crossing number of sufficiently dense graphs, can be significantly improved for dense graphs *with high girth* [PST00]. We hence wonder whether a similar improvement for computing crossing numbers exists.

Bibliography

[ADP19] Alan Arroyo, Martin Derka, and Irene Parada. "Extending Simple Drawings." In: *Proceedomgs of the 27th International Symposium on Graph Drawing and Network Visualization (GD 2019), Prague, Czech Republic*. Ed. by Daniel Archambault and Csaba D. Tóth. Vol. 11904. LNCS. Springer, 2019, pp. 230–243. DOI: 10.1007/978-3-030-35802-0_18.

[Alb76] Michael O. Albertson. "A lower bound for the independence number of a planar graph." In: *Journal of Combinatorial Theory, Series B* 20.1 (1976), pp. 84–93. DOI: 10.1016/0095-8956(76)90071-X.

[App+07] David L. Applegate, William J. Cook, Sanjeeb Dash, and Daniel G. Espinoza. "Exact solutions to linear programming problems." In: *Operations Research Letters* 35.6 (2007), pp. 693–699. DOI: 10.1016/j.orl.2006.12.010.

[App+09] David Applegate, Robert E. Bixby, Vasek Chvátal, William J. Cook, Daniel G. Espinoza, Marcos Goycoolea, and Keld Helsgaun. "Certification of an optimal TSP tour through 85, 900 cities." In: *Oper. Res. Lett.* 37.1 (2009), pp. 11–15. DOI: 10.1016/j.orl.2008.09.006.

[Arr+20] Alan Arroyo, Fabian Klute, Irene Parada, Raimund Seidel, Birgit Vogtenhuber, and Tilo Wiedera. "Inserting one edge into a simple drawing is hard." In: *Proceedings of the 46th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2020), Leeds, West Yorkshire, Great Britain*. LNCS. to appear. Springer, 2020.

[Bak94] Brenda S. Baker. "Approximation Algorithms for NP-Complete Problems on Planar Graphs." In: *Journal of the ACM* 41.1 (1994), pp. 153–180. DOI: 10.1145/174644.174650.

[Bar+10] Thomas Bartz-Beielstein, Marco Chiarandini, Luís Paquete, and Mike Preuss, eds. *Experimental Methods for the Analysis of Optimization Algorithms*. Springer, 2010. DOI: 10.1007/978-3-642-02538-9.

[Bey+16] Stephan Beyer, Markus Chimani, Ivo Hedtke, and Michal Kotrbčík. "A Practical Method for the Minimum Genus of a Graph: Models and Experiments." In: *Proceedings of the 15th International Symposium on Experimental Algorithms (SEA 2016), St. Petersburg, Russia*. Vol. 9685. LNCS. 2016, pp. 75–88. DOI: 10.1007/978-3-319-38851-9_6.

[BEZ15] Glencora Borradaile, David Eppstein, and Pingan Zhu. "Planar Induced Subgraphs of Sparse Graphs." In: *Journal of Graph Algorithms and Applications (JGAA)* 19.1 (2015), pp. 281–297. DOI: 10.7155/jgaa.00358.

[BKK07] Glencora Borradaile, Claire Kenyon-Mathieu, and Philip N. Klein. "A polynomial-time approximation scheme for Steiner tree in planar graphs." In: *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2007), New Orleans, Louisiana, USA*. SIAM, 2007, pp. 1285–1294. URL: http://dl.acm.org/citation.cfm?id=1283383.1283521.

[BKS17] Amariah Becker, Philip N. Klein, and David Saulpic. "A Quasi-Polynomial-Time Approximation Scheme for Vehicle Routing on Planar and Bounded-Genus Graphs." In: *Proceedings of the 25th Annual European Symposium on Algorithms (ESA 2017), Vienna, Austria*. Vol. 87. LIPIcs. 2017, pp. 12:1–12:15. DOI: 10.4230/LIPIcs.ESA.2017.12.

[Bok+a] Drago Bokal, Markus Chimani, Alexander Nover, Jöran Schierbaum, Tobias Stolzmann, Mirko H. Wagner, and Tilo Wiedera. *Properties of Large 2-Crossing-Critical Graphs*. Submitted in 2020.

[Bok+b] Drago Bokal, Zdeněk Dvořák, Petr Hliněný, Jesús Leaños, Bojan Mohar, and Tilo Wiedera. *Bounded Degree Conjecture Holds Precisely for c-Crossing-Critical Graphs with* $c \leq 12$. Journal version submitted in 2020.

[Bok+16] Drago Bokal, Bogdan Oporowski, R. Bruce Richter, and Gelasio Salazar. "Characterizing 2-crossing-critical graphs." In: *Advances in Applied Mathematics* 74 (2016), pp. 23–208. DOI: 10.1016/j.aam.2015.10.003.

[Bok+19a] Drago Bokal, Mojca Bracic, Marek Derňár, and Petr Hliněný. "On Degree Properties of Crossing-Critical Families of Graphs." In: *The Electronic Journal of Combinatorics* 26.1 (2019), P1.53. DOI: 10.37236/7753.

[Bok+19b] Drago Bokal, Zdeněk Dvořák, Petr Hliněný, Jesús Leaños, Bojan Mohar, and Tilo Wiedera. "Bounded Degree Conjecture Holds Precisely for c-Crossing-Critical Graphs with $c \leq 12$." In: *Proceedings of the 35th International Symposium on Computational Geometry (SoCG 2019), Portland, Oregon, USA*. Vol. 129. LIPIcs. 2019, pp. 14:1–14:15. DOI: 10.4230/LIPIcs.SoCG.2019.14.

[Bök+20] Fritz Bökler, Markus Chimani, Mirko H. Wagner, and Tilo Wiedera. "An Experimental Study of ILP Formulations for the Longest Induced Path Problem." In: *Proceedings of the 6th International Symposium on Combinatorial Optimization (ISCO 2020), Montreal, QC, Canada*. Vol. 12176. LNCS. Springer, 2020, pp. 89–101. DOI: 10.1007/978-3-030-53262-8_8.

[Bok10] Drago Bokal. "Infinite families of crossing-critical graphs with prescribed average degree and crossing number." In: *Journal of Graph Theory* 65.2 (2010), pp. 139–162. DOI: 10.1002/jgt.20470.

[Bri20] G. Brinkmann. *A Practical Algorithm for the Computation of the Genus*. 2020. arXiv: 2005.08243 [math.CO].

[BRS89] Matthew G. Brin, David E. Rauschenberg, and Craig C. Squier. "On the genus of the semidirect product of $\mathbb{Z}_9$ by $\mathbb{Z}_3$." In: *Journal of Graph Theory* 13.1 (1989), pp. 49–61. DOI: 10.1002/jgt.3190130108.

[BTT84] Carlo Batini, Maurizio Talamo, and Roberto Tamassia. "Computer aided layout of entity relationship diagrams." In: *Journal of Systems and Software* 4.2-3 (1984), pp. 163–173. DOI: 10.1016/0164-1212(84)90006-2.

[Buc+08] Christoph Buchheim, Markus Chimani, Dietmar Ebner, Carsten Gutwenger, Michael Jünger, Gunnar W. Klau, Petra Mutzel, and René Weiskircher. "A branch-and-cut approach to the crossing number problem." In: *Discrete Optimization* 5.2 (2008), pp. 373–388. DOI: 10.1016/j.disopt.2007.05.006.

[Cab13] Sergio Cabello. "Hardness of Approximation for Crossing Number." In: *Discrete & Computational Geometry* 49.2 (2013), pp. 348–358. DOI: 10.1007/s00454-012-9440-6.

[Căl+98] Gruia Călinescu, Cristina G. Fernandes, Ulrich Finkler, and Howard J. Karloff. "A Better Approximation Algorithm for Finding Planar Subgraphs." In: *Journal of Algorithms* 27.2 (1998), pp. 269–302. DOI: 10.1006/jagm.1997.0920.

[Cap+11] Alberto Caprara, Marcus Oswald, Gerhard Reinelt, Robert Schwarz, and Emiliano Traversi. "Optimal linear arrangements using betweenness variables." In: *Mathematical Programming Computation* 3.3 (2011), pp. 261–280. DOI: 10.1007/s12532-011-0027-7.

[CG09] Markus Chimani and Carsten Gutwenger. "Non-planar core reduction of graphs." In: *Discrete Mathematics* 309.7 (2009), pp. 1838–1855. DOI: 10.1016/j.disc.2007.12.078.

[CG12] Markus Chimani and Carsten Gutwenger. "Advances in the Planarization Method: Effective Multiple Edge Insertions." In: *Journal of Graph Algorithms and Applications (JGAA)* 16.3 (2012), pp. 729–757. DOI: 10.7155/jgaa.00264.

[CG15] Marston Conder and Ricardo Grande. "On Embeddings of Circulant Graphs." In: *The Electronic Journal of Combinatorics* 22.2 (2015), P2.28. DOI: 10.37236/4013.

[CGM10] Markus Chimani, Carsten Gutwenger, and Petra Mutzel. "Experiments on Exact Crossing Minimization Using Column Generation." In: *ACM Journal of Experimental Algorithmics* 14 (2010), pp. 4:3.4–4:3.18. ISSN: 1084-6654. DOI: 10.1145/1498698.1564504.

[CH16] Markus Chimani and Petr Hliněný. "Inserting Multiple Edges into a Planar Graph." In: *Proceedings of the 32nd International Symposium on Computational Geometry (SoCG 2016), Boston, MA, USA*. Vol. 51. LIPIcs. 2016, 30:1–30:15. DOI: 10.4230/LIPIcs.SoCG.2016.30.

[CH17] Markus Chimani and Petr Hliněný. "A tighter insertion-based approximation of the crossing number." In: *Journal of Combinatorial Optimization* 33.4 (2017), pp. 1183–1225. DOI: 10.1007/s10878-016-0030-z.

[Che+07] Jianer Chen, Iyad A. Kanj, Ljubomir Perkovic, Eric Sedgwick, and Ge Xia. "Genus characterizes the complexity of certain graph problems: Some tight results." In: *Journal of Computer and System Sciences* 73.6 (2007), pp. 892–907. DOI: 10.1016/j.jcss.2006.11.001.

[Che94] Jianer Chen. "A Linear-Time Algorithm for Isomorphism of Graphs of Bounded Average Genus." In: *SIAM Journal on Discrete Mathematics* 7.4 (1994), pp. 614–631. DOI: 10.1137/S0895480191196769.

[Chi+13] Markus Chimani, Carsten Gutwenger, Mike Juenger, Gunnar W. Klau, Karsten Klein, and Petra Mutzel. "The Open Graph Drawing Framework (OGDF)." In: *Handbook on Graph Drawing and Visualization*. Ed. by Roberto Tamassia. Chapman and Hall/CRC, 2013, pp. 543–569. ISBN: 978-1-5848-8412-5. URL: https://crcpress.com/Handbook-of-Graph-Drawing-and-Visualization/Tamassia/9781584884125.

[Chi+20] Markus Chimani, Christine Dahn, Martina Juhnke-Kubitzke, Nils M. Kriege, Petra Mutzel, and Alexander Nover. "Maximum Cut Parameterized by Crossing Number." In: *Journal of Graph Algorithms and Applications (JGAA)* 24.3 (2020), pp. 155–170. DOI: 10.7155/jgaa.00523.

[Chi08] Markus Chimani. "Computing Crossing Numbers." PhD thesis. TU Dortmund, 2008.

[Chi11] Markus Chimani. "Facets in the Crossing Number Polytope." In: *SIAM Journal on Discrete Mathematics* 25.1 (2011), pp. 95–111. DOI: 10.1137/09076965X.

[CHN19] Kieran Clancy, Michael Haythorpe, and Alex Newcombe. "An effective crossing minimisation heuristic based on star insertion." In: *Journal of Graph Algorithms and Applications (JGAA)*. 23.2 (2019), pp. 135–166. DOI: 10.7155/jgaa.00487.

[CHW16] Markus Chimani, Ivo Hedtke, and Tilo Wiedera. "Limits of Greedy Approximation Algorithms for the Maximum Planar Subgraph Problem." In: *Proceedings of the 27th International Workshop on Combinatorial Algorithms (IWOCA 2016), Helsinki, Finland*. Vol. 9843. LNCS, 2016, pp. 334–346. DOI: 10.1007/978-3-319-44543-4_26.

[CHW18] Markus Chimani, Ivo Hedtke, and Tilo Wiedera. "Exact Algorithms for the Maximum Planar Subgraph Problem: New Models and Experiments." In: *Proceedings of the 17th International Symposium on Experimental Algorithms (SEA 2018), L'Aquila, Italy*. Vol. 103. LIPIcs. 2018, 22:1–22:15. DOI: 10.4230/LIPIcs.SEA.2018.22.

[CHW19] Markus Chimani, Ivo Hedtke, and Tilo Wiedera. "Exact Algorithms for the Maximum Planar Subgraph Problem: New Models and Experiments." In: *ACM Journal of Experimental Algorithmics* 24.1 (2019). DOI: 10.1145/3320344.

[Cim95a] Robert J. Cimikowski. "An Analysis of Some Heuristics for the Maximum Planar Subgraph Problem." In: *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA 1995), San Francisco, California, USA*. ACM/SIAM, 1995, pp. 322–331. URL: http://dl.acm.org/citation.cfm?id=313651.313713.

[Cim95b] Robert J. Cimikowski. "On Heuristics for Determining the Thickness of a Graph." In: *Information Sciences* 85.1-3 (1995), pp. 87–98. DOI: 10.1016/0020-0255(95)00011-D.

[CKW16] Markus Chimani, Karsten Klein, and Tilo Wiedera. "A Note on the Practicality of Maximal Planar Subgraph Algorithms." In: *Proceedings of the 24th International Symposium on Graph Drawing and Network Visualization, (GD 2016), Athens, Greece*. Vol. 9801. LNCS, 2016, pp. 357–364. DOI: 10.1007/978-3-319-50106-2_28.

[CL91] Jason Cong and C.L. Liu. "On the k-layer planar subset and topological via minimization problems." In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 10.8 (1991), pp. 972–981. DOI: 10.1109/43.85735.

[CM13] Sergio Cabello and Bojan Mohar. "Adding one edge to planar graphs makes crossing number and 1-planarity hard." In: *SIAM Journal on Computing* 42 (5 2013), pp. 1803–1829.

[CMB08] Markus Chimani, Petra Mutzel, and Immanuel Bomze. "A New Approach to Exact Crossing Minimization." In: *Proceedings of the 16th Annual European Symposium on Algorithms (ESA 2008), Karlsruhe, Germany*. Vol. 5193. LNCS. 2008, pp. 284–296. DOI: 10.1007/978-3-540-87744-8_24.

[CMS07] Markus Chimani, Petra Mutzel, and Jens M. Schmidt. "Efficient Extraction of Multiple Kuratowski Subdivisions." In: *Proceedings 15th International Symposium on Graph Drawing (GD 2007), Sydney, Australia*. Vol. 4875. LNCS, 2007, pp. 159–170. DOI: 10.1007/978-3-540-77537-9_17.

[CS17] Parinya Chalermsook and Andreas Schmid. "Finding Triangles for Maximum Planar Subgraphs." In: *Proceedings of the 11th International Conference and Workshops on Algorithms and Computation (WALCOM 2017), Hsinchu, Taiwan*. Vol. 10167. LNCS. 2017, pp. 373–384. DOI: 10.1007/978-3-319-53925-6_29.

[CS18] Chandra Chekuri and Anastasios Sidiropoulos. "Approximation Algorithms for Euler Genus and Related Problems." In: *SIAM Journal on Computing* 47.4 (2018), pp. 1610–1643. DOI: 10.1137/14099228X.

[CTW] Markus Chimani, Niklas Troost, and Tilo Wiedera. *Approximating Multistage Matching Problems*. Submitted in 2020. arXiv: 2002.06887.

[CW16] Markus Chimani and Tilo Wiedera. "An ILP-based Proof System for the Crossing Number Problem." In: *Proceedings of the 24th Annual European Symposium on Algorithms (ESA 2016), Aarhus, Denmark*. Vol. 57. LIPIcs, 2016, pp. 29:1–29:13. DOI: 10.4230/LIPIcs.ESA.2016.29.

[CW18] Markus Chimani and Tilo Wiedera. "Cycles to the Rescue! Novel Constraints to Compute Maximum Planar Subgraphs Fast." In: *Proceedings of the 26th Annual European Symposium on Algorithms, (ESA 2018), Helsinki, Finland*. Vol. 112. LIPIcs, 2018, pp. 19:1–19:14. DOI: 10.4230/LIPIcs.ESA.2018.19.

[CW19] Markus Chimani and Tilo Wiedera. "Stronger ILPs for the Graph Genus Problem." In: *Proceedings of the 27th Annual European Symposium on Algorithms (ESA 2019), Munich/Garching, Germany*. Vol. 144. LIPIcs. 2019, pp. 30:1–30:15. DOI: 10.4230/LIPIcs.ESA.2019.30.

[Dhi+03] Marcel Dhiflaoui, Stefan Funke, Carsten Kwappik, Kurt Mehlhorn, Michael Seel, Elmar Schömer, Ralph Schulte, and Dennis Weber. "Certifying and repairing solutions to large LPs how good are LP-solvers?" In: *Proceedings of the 14th Annual ACM-SIAM*

Symposium on Discrete Algorithms (SODA 2003), Baltimore, Maryland, USA. ACM/SIAM, 2003, pp. 255–256.

[DHM18] Zdeněk Dvořák, Petr Hliněný, and Bojan Mohar. "Structure and Generation of Crossing-Critical Graphs." In: *Proceedings of the 34th International Symposium on Computational Geometry (SOCG 2018), Budapest, Hungary*. Ed. by Bettina Speckmann and Csaba D. Tóth. Vol. 99. LIPIcs. 2018, pp. 33:1–33:14. DOI: 10.4230/LIPIcs.SoCG.2018.33.

[DHT06] Erik D. Demaine, MohammadTaghi Hajiaghayi, and Dimitrios M. Thilikos. "The Bidimensional Theory of Bounded-Genus Graphs." In: *SIAM Journal on Discrete Mathematics* 20.2 (2006), pp. 357–371. DOI: 10.1137/040616929.

[Di +97] Giuseppe Di Battista, Ashim Garg, Giuseppe Liotta, Roberto Tamassia, Emanuele Tassinari, and Francesco Vargiu. "An experimental comparison of four graph drawing algorithms." In: *Computational Geometry. Theory and Applications* 7.5-6 (1997). 11th ACM Symposium on Computational Geometry (Vancouver, BC, 1995), pp. 303–325. DOI: 10.1016/S0925-7721(96)00005-3.

[Die12] Reinhard Diestel. *Graph Theory, 4th Edition*. Vol. 173. Graduate texts in mathematics. Springer, 2012. ISBN: 978-3-642-14278-9.

[DM10] Zdeněk Dvořák and Bojan Mohar. "Crossing-critical graphs with large maximum degree." In: *Journal of Combinatorial Theory, Series B* 100.4 (2010), pp. 413–417. DOI: 10.1016/j.jctb.2009.11.003.

[DM41] Ben Dushnik and E. W. Miller. "Partially Ordered Sets." In: *American Journal of Mathematics* 63.3 (1941), pp. 600–610. URL: http://www.jstor.org/stable/2371374.

[EF01] Keith Edwards and Graham Farr. "An Algorithm for Finding Large Induced Planar Subgraphs." In: *Proceedings of the 9th International Symposium on Graph Drawing (GD 2001), Vienna, Austria*. Vol. 2265. LNCS. 2001, pp. 75–83. DOI: 10.1007/3-540-45848-4_6.

[EF08] Keith Edwards and Graham Farr. "Planarization and fragmentability of some classes of graphs." In: *Discrete Mathematics* 308.12 (2008), pp. 2396–2406. DOI: 10.1016/j.disc.2007.05.007.

[EFF04] John A. Ellis, Hongbing Fan, and Michael R. Fellows. "The dominating set problem is fixed parameter tractable for graphs of bounded genus." In: *Journal of Algorithms* 52.2 (2004), pp. 152–168. DOI: 10.1016/j.jalgor.2004.02.001.

[Erl01] Thomas Erlebach. "Approximation Algorithms and Complexity Results for Path Problems in Trees of Rings." In: *Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science (MFCS 2001), Marianske Lazne, Czech Republic*. Vol. 2136. LNCS. 2001, pp. 351–362. DOI: 10.1007/3-540-44683-4_31.

[Far+04] Luérbio Faria, Celina M. H. de Figueiredo, Sylvain Gravier, Candido Ferreira Xavier de Mendonça, and Jorge Stolfi. "Nonplanar vertex deletion: maximum degree thresholds for NP/MaxSNP-hardness and a I-approximation for finding maximum planar induced subgraphs." In: *Electronic Notes in Discrete Mathematics* 18 (2004), pp. 121–126. DOI: 10.1016/j.endm.2004.06.019.

[Far+06] Luérbio Faria, Celina M. H. de Figueiredo, Sylvain Gravier, Candido Ferreira Xavier de Mendonça, and Jorge Stolfi. "On maximum planar induced subgraphs." In: *Discrete Applied Mathematics* 154.13 (2006), pp. 1774–1782. DOI: 10.1016/j.dam.2006.03.021.

[FC07] Cristina G. Fernandes and Gruia Călinescu. "Maximum Planar Subgraph." In: *Handbook of Approximation Algorithms and Metaheuristics*. Chapman & Hall/CRC, 2007. DOI: 10.1201/9781420010749.ch56.

[FFM01] Luérbio Faria, Celina M. H. de Figueiredo, and Candido Ferreira Xavier de Mendonça. "SPLITTING NUMBER is NP-complete." In: *Discrete Applied Mathematics* 108.1-2 (2001), pp. 65–83.

[FFM04] Luérbio Faria, Celina M. H. de Figueiredo, and Candido Ferreira Xavier de Mendonça. "On the complexity of the approximation of nonplanarity parameters for cubic graphs." In: *Discrete Applied Mathematics* 141.1-3 (2004), pp. 119–134. DOI: 10.1016/S0166-218X(03)00370-6.

[FGG85] Les R. Foulds, Peter B. Gibbons, and J. W. Giffin. "Facilities Layout Adjacency Determination: An Experimental Comparison of Three Graph Theoretic Heuristics." In: *Operations Research* 33.5 (1985), pp. 1091–1106. DOI: 10.1287/opre.33.5.1091.

[FR76] L. R. Foulds and D. F. Robinson. "A strategy for solving the plant layout problem." In: *Operational Research Quarterly* 27.4, part 1 (1976), pp. 845–855. DOI: 10.2307/3009168.

[FR85] Hubert de Fraysseix and Pierre Rosenstiehl. "A characterization of planar graphs by Trémaux orders." In: *Combinatorica* 5.2 (1985), pp. 127–135. DOI: 10.1007/BF02579375.

[FST18] Manuel Fernández, Nicholas Sieger, and Michael Tait. "Maximal Planar Subgraphs of Fixed Girth in Random Graphs." In: *The Electronic Journal of Combinatorics* 25.2 (2018), P2.45. DOI: 10.37236/7114.

[FTV11] Fedor V. Fomin, Ioan Todinca, and Yngve Villanger. "Proceedings of the 19th Annual European Symposium on Algorithms (ESA 2011), Saarbrücken, Germany." In: vol. 6942. LNCS. 2011, pp. 287–298. DOI: 10.1007/978-3-642-23719-5_25.

[Geb+11] Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Marius Thomas Schneider. "Potassco: The Potsdam Answer Set Solving Collection." In: *AI Communications* 24.2 (2011), pp. 107–124. DOI: 10.3233/AIC-2011-0491.

[GHT84] John R. Gilbert, Joan P. Hutchinson, and Robert Endre Tarjan. "A Separator Theorem for Graphs of Bounded Genus." In: *Journal of Algorithms* 5.3 (1984), pp. 391–407. DOI: 10.1016/0196-6774(84)90019-1.

[GJ79] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. ISBN: 0-7167-1044-7.

[Gle+18] Ambros Gleixner, Michael Bastubbe, Leon Eifler, Tristan Gally, Gerald Gamrath, Robert Lion Gottwald, Gregor Hendel, Christopher Hojny, Thorsten Koch, Marco E. Lübbecke, Stephen J. Maher, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Franziska Schlösser, Christoph Schubert, Felipe Serrano, Yuji Shinano, Jan Merlin Viernickel, Matthias Walter, Fabian Wegscheider, Jonas T. Witt, and Jakob Witzig. *The SCIP Optimization Suite 6.0*. Technical Report. Optimization Online, 2018. URL: http://www.optimization-online.org/DB_HTML/2018/07/6692.html.

[GM00] Carsten Gutwenger and Petra Mutzel. "A Linear Time Implementation of SPQR-Trees." In: *Proceedings of the 8th International Symposium on Graph Drawing (GD 2000), Colonial Williamsburg, VA, USA*. Vol. 1984. LNCS. Springer, 2000, pp. 77–90. DOI: 10.1007/3-540-44541-2_8.

[GM03] Carsten Gutwenger and Petra Mutzel. "An Experimental Study of Crossing Minimization Heuristics." In: *Proceedings of the 11th International Symposium on Graph Drawing (GD 2003), Perugia, Italy*. Vol. 2912. LNCS. 2003, pp. 13–24. DOI: 10.1007/978-3-540-24595-7_2.

[Gol63] A.J. Goldstein. "An efficient and constructive algorithm for testing whether a graph can be embedded in a plane." In: *Graph and Combinatorics Conference*. 1963.

[GW92] Harold N. Gabow and Herbert H. Westermann. "Forests, Frames, and Games: Algorithms for Matroid Sums and Applications." In: *Algorithmica* 7.5–6 (1992), pp. 465–497. DOI: 10.1007/BF01758774.

[Har69] F. Harary. *Graph Theory*. Addison-Wesley series in mathematics. Addison-Wesley Publishing Company, 1969. ISBN: 978-0201410334.

[Hed17] Ivo Hedtke. "Minimum Genus and Maximum Planar Subgraph: Exact Algorithms and General Limits of Approximation Algorithms." PhD thesis. Osnabrück University, 2017. URL: https://repositorium.ub.uos.de/handle/urn:nbn:de:gbv:700-2017082416212.

[Hen+97] Monika Rauch Henzinger, Philip N. Klein, Satish Rao, and Sairam Subramanian. "Faster Shortest-Path Algorithms for Planar Graphs." In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 3–23. DOI: 10.1006/jcss.1997.1493.

[Hli03] Petr Hliněný. "Crossing-number critical graphs have bounded path-width." In: *Journal of Combinatorial Theory, Series B* 88.2 (2003), pp. 347–367. DOI: 10.1016/S0095-8956(03)00037-6.

[Hli06] Petr Hliněný. "Crossing number is hard for cubic graphs." In: *Journal of Combinatorial Theory, Series B* 96.4 (2006), pp. 455–471. DOI: 10.1016/j.jctb.2005.09.009.

[HS11] Steven Homer and Alan L. Selman. *Computability and Complexity Theory, Second Edition*. Texts in Computer Science. Springer, 2011. DOI: 10.1007/978-1-4614-0682-2.

[HST12] César Hernández-Vélez, Gelasio Salazar, and Robin Thomas. "Nested cycles in large triangulations and crossing-critical graphs." In: *Journal of Combinatorial Theory, Series B* 102.1 (2012), pp. 86–92. DOI: 10.1016/j.jctb.2011.04.006.

[Hsu05] Wen-Lian Hsu. "A Linear Time Algorithm for Finding a Maximal Planar Subgraph Based on PC-Trees." In: *Proceedings of the 11th International Computing and Combinatorics Conference, (COCOON 2005), Kunming, China*. Vol. 3595. LNCS. 2005, pp. 787–797. DOI: 10.1007/11533719_80.

[HT71] John E. Hopcroft and Robert Endre Tarjan. "A V^2 Algorithm for Determining Isomorphism of Planar Graphs." In: *Information Processing Letters* 1.1 (1971), pp. 32–34. DOI: 10.1016/0020-0190(71)90019-6.

[HW07] Jan M. Hochstein and Karsten Weihe. "Maximum s-t-flow with k crossings in $O(k^3 n \log n)$ time." In: *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2007), New Orleans, Louisiana, USA*. Ed. by Nikhil Bansal, Kirk Pruhs, and Clifford Stein. SIAM, 2007, pp. 843–847. URL: http://dl.acm.org/citation.cfm?id=1283383.1283473.

[JM96] Michael Jünger and Petra Mutzel. "Maximum Planar Subgraphs and Nice Embeddings: Practical Layout Tools." In: *Algorithmica* 16.1 (1996), pp. 33–59. DOI: 10.1007/BF02086607.

[Joh04] John M. Boyer and Wendy J. Myrvold. "On the Cutting Edge: Simplified O(n) Planarity by Edge Addition." In: *Journal of Graph Algorithms and Applications (JGAA)* 8.3 (2004), pp. 241–273. DOI: 10.7155/jgaa.00091.

[JT00] Michael Jünger and Stefan Thienel. "The ABACUS system for branch-and-cut-and-price algorithms in integer programming and combinatorial optimization." In: *Software: Practice and Experience* 30.11 (2000), pp. 1325–1352. DOI: 10.1002/1097-024X(200009)30:11<1325::AID-SPE342>3.0.CO;2-T.

[JTS89] R. Jayakumar, Krishnaiyan Thulasiraman, and M. N. S. Swamy. "$O(n^2)$ algorithms for graph planarization." In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 8.3 (1989), pp. 257–267. DOI: 10.1109/43.21845.

[Kan92] Goos Kant. *An $O(n^2)$ maximal planarization algorithm based on PQ-trees*. Technical Report RUU-CS-92-03. P.O. Box 80.089, 3508 TB Utrecht, the Netherlands: Department of Computer Science, Utrecht University, 1992.

[Kle08] Philip N. Klein. "A Linear-Time Approximation Scheme for TSP in Undirected Planar Graphs with Edge-Weights." In: *SIAM Journal on Computing* 37.6 (2008), pp. 1926–1952. DOI: 10.1137/060649562.

[KMR08] K. Kawarabayashi, B. Mohar, and B. Reed. "A Simpler Linear Time Algorithm for Embedding Graphs into an Arbitrary Surface and the Genus of Graphs of Bounded Tree-Width." In: *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008)*. IEEE, 2008, pp. 771–780. DOI: 10.1109/FOCS.2008.53.

[KMV00] T. Koch, A. Martin, and S. Voß. *SteinLib: An Updated Library on Steiner Tree Problems in Graphs*. Technical Report ZIB-Report 00-37. Takustr. 7, Berlin: Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2000. URL: http://elib.zib.de/steinlib.

[KR02] A. Kaveh and H. Rahami. "An Efficient Algorithm for Embedding Nonplanar Graphs in Planes." In: *Journal of Mathematical Modelling and Algorithms* 1.4 (2002), pp. 257–268. DOI: 10.1023/A:1021616706461.

[KR07] Ken-ichi Kawarabayashi and Bruce A. Reed. "Computing crossing number in linear time." In: *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC 2007), San Diego, California, USA*. 2007, pp. 382–390. DOI: 10.1145/1250790.1250848.

[Kra91] Jan Kratochvíl. "String graphs. II. recognizing string graphs is NP-hard." In: *Journal of Combinatorial Theory, Series B* 52.1 (1991), pp. 67–78. DOI: 10.1016/0095-8956(91)90091-W.

[KS15] Ken-ichi Kawarabayashi and Anastasios Sidiropoulos. "Beyond the Euler Characteristic: Approximating the Genus of General Graphs." In: *Proceedings of the 47th Annual ACM on Symposium on Theory of Computing (STOC 2015), Portland, OR, USA*. 2015, pp. 675–682. DOI: 10.1145/2746539.2746583.

[Kur30] Kasimir Kuratowski. "Sur le problème des courbes gauches en topologie." In: *Fundamenta Mathematicae* 15 (1930), pp. 271–283.

[KY03] Sinichiro Kawano and Koichi Yamazaki. "Worst case analysis of a greedy algorithm for graph thickness." In: *Information Processing Letters* 85.6 (2003), pp. 333–337. DOI: 10.1016/S0020-0190(02)00432-5.

[Kyn+15] Jan Kynčl, János Pach, Radoš Radoičićd, and Géza Tóthb. "Saturated simple and k-simple topological graphs." In: *Computational Geometry* 48.4 (2015), pp. 295–310. DOI: 10.1016/j.comgeo.2014.10.008.

[LG79] P. C. Liu and R. C. Geldmacher. "On the deletion of nonplanar edges of a graph." In: *Proceeidngs of the 10th Southeastern Conference on Combinatorics, Graph Theory and Computing (SEICCGTC 1979), Boca Raton, Florida, USA*. Congress. Numer., XXIII–XXIV. Utilitas Mathematica, 1979, pp. 727–738.

[Lie01] Annegret Liebers. "Planarizing Graphs - A Survey and Annotated Bibliography." In: *Journal of Graph Algorithms and Applications (JGAA)* 5.1 (2001). DOI: 10.7155/jgaa.00032.

[LT80] Richard J. Lipton and Robert Endre Tarjan. "Applications of a Planar Separator Theorem." In: *SIAM Journal on Computing* 9.3 (1980), pp. 615–627. DOI: 10.1137/0209046.

[Man83] Anthony Mansfield. "Determining the thickness of graphs is NP-hard." In: *Mathematical Proceedings of the Cambridge Philosophical Society* 93.1 (1983), pp. 9–23. DOI: 10.1017/S030500410006028X.

[MF07] Kerri Morgan and Graham Farr. "Approximation Algorithms for the Maximum Induced Planar and Outerplanar Subgraph Problems." In: *Journal of Graph Algorithms and Applications (JGAA)* 11.1 (2007), pp. 165–193. DOI: 10.7155/jgaa.00141.

[MK11] Wendy J. Myrvold and William L. Kocay. "Errors in graph embedding algorithms." In: *Journal of Computer and System Sciences (JCSS)* 77.2 (2011), pp. 430–438. DOI: 10.1016/j.jcss.2010.06.002.

[MNS17] Yury Makarychev, Amir Nayyeri, and Anastasios Sidiropoulos. "A Pseudo-Approximation for the Genus of Hamiltonian Graphs." In: *Theory of Computing* 13.1 (2017), pp. 1–47. DOI: 10.4086/toc.2017.v013a005.

[MNW07] Bojan Mohar, Richard J. Nowakowski, and Douglas B. West. "Research problems from the 5th Slovenian Conference (Bled, 2003)." In: *Discrete Mathematics* 307.3-5 (2007), pp. 650–658. DOI: 10.1016/j.disc.2006.07.013.

[Moh99] Bojan Mohar. "A Linear Time Algorithm for Embedding Graphs in an Arbitrary Surface." In: *SIAM Journal on Discrete Mathematics* 12.1 (1999), pp. 6–26. DOI: 10.1137/S089548019529248X.

[MOS98] Petra Mutzel, Thomas Odenthal, and Mark Scharbrodt. "The Thickness of Graphs: A Survey." In: *Graphs and Combinatorics* 14.1 (1998), pp. 59–73. DOI: 10.1007/PL00007219.

[MP12] Erkki Mäkinen and Timo Poranen. "An Annotated Bibliography on the Thickness, Outerthickness, and Arboricity of a Graph." In: *Missouri Journal of Mathematical Sciences* 24.1 (2012), pp. 76–87. DOI: 10.35834/mjms/1337950501.

[MP93] Matthias Middendorf and Frank Pfeiffer. "On the complexity of the disjoint paths problems." In: *Combinatorica* 13.1 (1993), pp. 97–107. DOI: 10.1007/BF01202792.

[MPP18] Dániel Marx, Marcin Pilipczuk, and Michal Pilipczuk. "On Subexponential Parameterized Algorithms for Steiner Tree and Directed Subset TSP on Planar Graphs." In: *Proceedings of the 59th Annual Symposium on Foundations of Computer Science (FOCS 2018), Paris, France*. IEEE, 2018, pp. 474–484. DOI: 10.1109/FOCS.2018.00052.

[MPV01] Erkki Mäkinen, Timo Poranen, and Petri Vuorenmaa. "A genetic algorithm for determining the thickness of a graph." In: *Information Sciences* 138.1-4 (2001), pp. 155–164. DOI: 10.1016/S0020-0255(01)00126-8.

[MPW05] Dragan Marusic, Tomaz Pisanski, and Steve Wilson. "The genus of the GRAY graph is 7." In: *European Journal of Combinatorics* 26.3-4 (2005), pp. 377–385. DOI: 10.1016/j.ejc.2004.01.015.

[MS12] Dániel Marx and Ildikó Schlotter. "Obtaining a Planar Graph by Vertex Deletion." In: *Algorithmica* 62.3-4 (2012), pp. 807–822. DOI: 10.1007/s00453-010-9484-z.

[MT01] Bojan Mohar and Carsten Thomassen. *Graphs on Surfaces*. series in the mathematical sciences. Johns Hopkins University Press, 2001. ISBN: 978-0-8018-6689-0.

[MU05] Michael Mitzenmacher and Eli Upfal. *Probability and computing - randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005. ISBN: 978-0-521-83540-4.

[Mut94] Petra Mutzel. "The maximum planar subgraph problem." PhD thesis. Köln University, 1994.

[Nor95] Stephen C. North. *5114 directed graphs*. Manuscript. 1995.

[NRM14] Lars Noschinski, Christine Rizkallah, and Kurt Mehlhorn. "Verification of Certifying Computations through AutoCorres and Simpl." In: *Proceedings of the 6th NASA International Symposium on Formal Methods (NFM 2014), Houston, TX, USA*. Ed. by Julia M. Badger and Kristin Yvonne Rozier. Vol. 8430. LNCS. Springer, 2014, pp. 46–61. DOI: 10.1007/978-3-319-06200-6_4.

[NVZ01] Takao Nishizeki, Jens Vygen, and Xiao Zhou. "The edge-disjoint paths problem is NP-complete for series–parallel graphs." In: *Discrete Applied Mathematics* 115.1-3 (2001), pp. 177–186. DOI: 10.1016/S0166-218X(01)00223-2.

[OMa+] Joshua O'Madadhain, Danyel Fisher, Tom Nelson, Scott White, and Yan-Biao Boey. *BarabasiAlbertGenerator of the Java Universal Network/Graph Framework*. accessed in 2016. URL: http://jung.sourceforge.net.

[Por05] Timo Poranen. "A simulated annealing algorithm for determining the thickness of a graph." In: *Information Sciences* 172.1-2 (2005), pp. 155–172. DOI: 10.1016/j.ins.2004.02.029.

[Por08] Timo Poranen. "Two New Approximation Algorithms for the Maximum Planar Subgraph Problem." In: *Acta Cybernetica* 18.3 (2008), pp. 503–527.

[PR03] Benny Pinontoan and R. Bruce Richter. "Crossing numbers of sequences of graphs II: Planar tiles." In: *Journal of Graph Theory* 42.4 (2003), pp. 332–341. DOI: 10.1002/jgt.10097.

[PST00] János Pach, Joel Spencer, and Géza Tóth. "New Bounds on Crossing Numbers." In: *Discrete & Computational Geometry* 24.4 (2000), pp. 623–644. DOI: 10.1007/s004540010011.

[RM09] Olivier Roussel and Vasco M. Manquinho. "Pseudo-Boolean and Cardinality Constraints." In: *Handbook of Satisfiability*. Vol. 185. Frontiers in Artificial Intelligence and Applications. IOS Press, 2009, pp. 695–733. DOI: 10.3233/978-1-58603-929-5-695.

[RS04] Neil Robertson and Paul D. Seymour. "Graph Minors. XX. Wagner's conjecture." In: *Journal of Combinatorial Theory, Series B* 92.2 (2004), pp. 325–357. DOI: 10.1016/j.jctb.2004.08.001.

[RS90] Neil Robertson and Paul D. Seymour. "Graph minors. VIII. A Kuratowski theorem for general surfaces." In: *Journal of Combinatorial Theory, Series B* 48.2 (1990), pp. 255–288. DOI: 10.1016/0095-8956(90)90121-F.

[RS95] Neil Robertson and Paul D. Seymour. "Graph Minors. XIII. The Disjoint Paths Problem." In: *Journal of Combinatorial Theory, Series B* 63.1 (1995), pp. 65–110. DOI: 10.1006/jctb.1995.1006.

[RT93] R. Bruce Richter and Carsten Thomassen. "Minimal Graphs with Crossing Number at Least k." In: *Journal of Combinatorial Theory, Series B* 58.2 (1993), pp. 217–224. DOI: 10.1006/jctb.1993.1038.

[Sch13] Marcus Schaefer. "The Graph Crossing Number and its Variants: A Survey." In: *Electronic Journal of Combinatorics* DS21: Feb 14, 2020 (2013). DOI: https://doi.org/10.37236/2713.

[Sch18] Marcus Schaefer. *Crossing Numbers of Graphs*. 1st ed. CRC Press, 2018. DOI: 10.1201/9781315152394.

[Sch89] Walter Schnyder. "Planar graphs and poset dimension." In: *Order* 5.4 (1989), pp. 323–343. ISSN: 0167-8094. DOI: 10.1007/BF00353652.

[Sch99] Alexander Schrijver. *Theory of linear and integer programming*. series in discrete mathematics and optimization. Wiley, 1999. ISBN: 978-0-471-98232-6.

[SH99] Wei-Kuan Shih and Wen-Lian Hsu. "A New Planarity Test." In: *Theoretical Computer Science* 223.1-2 (1999), pp. 179–191. DOI: 10.1016/S0304-3975(98)00120-0.

[Sir84] Jozef Sirán. "Infinite families of crossing-critical graphs with a given crossing number." In: *Discrete Mathematics* 48.1 (1984), pp. 129–132. DOI: 10.1016/0012-365X(84)90140-7.

[SW99] Angelika Steger and Nicholas C. Wormald. "Generating Random Regular Graphs Quickly." In: *Combinatorics, Probability & Computing* 8.4 (1999), pp. 377–396.

[Szp30] Edward Szpilrajn. "Sur l'extension de l'ordre partiel." In: *Fundamenta Mathematicae* 16.1 (1930), pp. 386–389. URL: http://eudml.org/doc/212499.

[Tar95] Gaston Tarry. "Le problème des labyrinthes." In: *Nouvelles annales de mathématiques: journal des candidats aux écoles polytechnique et normale, Série 3* 14 (1895), pp. 187–190. URL: http://numdam.org/article/NAM_1895_3_14__187_1.pdf.

[TDB88] Roberto Tamassia, Giuseppe Di Battista, and Carlo Batini. "Automatic graph drawing and readability of diagrams." In: *IEEE Transactions on Systems, Man, and Cybernetics* 18.1 (1988), pp. 61–79. DOI: 10.1109/21.87055.

[Tho89] Carsten Thomassen. "The graph genus problem is NP-complete." In: *Journal of Algorithms* 10.4 (1989), pp. 568–576. DOI: 10.1016/0196-6774(89)90006-0.

[Tho90] Carsten Thomassen. "Embeddings of graphs with no short non-contractible cycles." In: *Journal of Combinatorial Theory, Series B* 48.2 (1990), pp. 155–177. DOI: 10.1016/0095-8956(90)90115-G.

[Tut63] William T. Tutte. "The Thickness of a Graph." In: *Indagationes Mathematicae* 25.2 (1963), pp. 567–577.

[Vrt14] Imrich Vrt'o. *Crossing Numbers of Graphs: A Bibliography*. 2014. URL: http://ftp.ifi.savba.sk/pub/imrich/crobib.pdf.

[VWM15] Yakir Vizel, Georg Weissenbacher, and Sharad Malik. "Boolean Satisfiability Solvers and Their Applications in Model Checking." In: *Proceedings of the IEEE* 103.11 (2015), pp. 2021–2035. DOI: 10.1109/JPROC.2015.2455034.

[Vyg95] Jens Vygen. "NP-completeness of Some Edge-disjoint Paths Problems." In: *Discrete Applied Mathematics* 61.1 (1995), pp. 83–90. DOI: 10.1016/0166-218X(93)E0177-Z.

[Wag37] K. Wagner. "Über eine Eigenschaft der ebenen Komplexe." In: *Mathematische Annalen* 114.1 (1937), pp. 570–590. DOI: 10.1007/BF01594196.

[WAN83] Toshimasa Watanabe, Tadashi Ae, and Akira Nakamura. "On the NP-hardness of edge-deletion and -contraction problems." In: *Discrete Applied Mathematics* 6.1 (1983), pp. 63–78. DOI: 10.1016/0166-218X(83)90101-4.

[Yan78] Mihalis Yannakakis. "Node- and Edge-Deletion NP-Complete Problems." In: *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC 1978), San Diego, California, USA*. 1978, pp. 253–264. DOI: 10.1145/800133.804355.

[ZM02] Lintao Zhang and Sharad Malik. "The Quest for Efficient Boolean Satisfiability Solvers." In: *Proceedings of the 18th International Conference on Automated Deduction (CADE-18), Copenhagen, Denmark*. Ed. by Andrei Voronkov. Vol. 2392. LNCS. 2002, pp. 295–313. DOI: 10.1007/3-540-45620-1_26.

List of Figures

1.1 Different drawings of the same graph. 10
1.2 Planar graph and its dual. 12

2.1 Performance of MPS heuristics. 36
2.2 Maximal planar subgraphs that are almost trees. 37
2.3 Constructions for the proof of Theorem 2.3.2. 39
2.4 Construction for the proof of Theorem 2.3.3. 40
2.5 Constructions for the proofs of Theorems 2.3.4 and 2.3.7. . . 41
2.6 Arising $K_{3,3}$-subdivisions. 42
2.7 Constructions in the proof of Theorem 2.3.9. 44
2.8 Construction in the proof of Theorem 2.3.10. 46
2.9 Construction for proof of Theorem 2.3.13. 49
2.10 Schematics of T-alike and T-opposite relations. 58
2.11 Skewness of random generated instance set SkewBound. . . 62
2.12 Success rate and running time on Rome graphs. 64
2.13 Success rate and running time on North graphs. 65
2.14 Comparison of MPS models' bounds on Rome graphs. 66
2.15 Comparison of MPS models' bounds on North graphs. . . . 67
2.16 Strength of the generalized Euler constraints. 72
2.17 Strength of the pseudo-tree model. 78
2.18 Strength of cycle-edge constraints. 80
2.19 Strength of two-cycles-path constraints. 82
2.20 Relation of paths w.r.t. a cycle. 84
2.21 Strength of Kuratowski-cycle constraints. 85
2.22 Performance of MPS models. 92

3.1 Performance of genus models. 107
3.2 Bounds identified by genus models. 109
3.3 Dual bounds and generated variables for genus model. 109

4.1 Performance of column generation for crossing number proofs. 126
4.2 User Interface for Retrieving Proofs of Crossing Numbers . . 127
4.3 The graph G_{13}^k of Definition 4.2.2, drawn with 13 crossings. . 129
4.4 Two cases of the induction step in the proof of Lemma 4.2.5. 132
4.5 Two drawings of the graph G_{13}^k. 133
4.6 Two more drawings of the graph G_{13}^k. 135
4.7 Application of a 4-to-3-expansion. 136
4.8 Construction of the corridor (b) from a known drawing (a). . 141
4.9 Different gadgets used in our construction. 143
4.10 Placement of gadgets in the corridor. 145
4.11 Transformation to a matching. 145

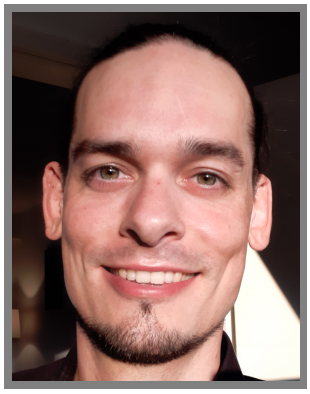
List of Tables

1.1 Overview of common identifiers. 3

2.1 Solution quality of MPS heuristics. 35
2.2 Running time of MPS heuristics. 35
2.3 Solved instances by MPS models. 61
2.4 Dual bounds by MPS models. 62
2.5 Solved instances by MPS models on SteinLib. 63
2.6 Solved instances by MPS models on Regular. 63
2.7 Solved instances by MPS models on an artificial bounded set. 68
2.8 Notation to encode algorithmic variants for MPS. 88
2.9 Performance for algorithmic variants for MPS. 89
2.10 Improvement of MPS algorithms over state-of-the-art. 90
2.11 Generated variables in MPS models. 90

3.1 Performance of genus algorithms. 108
3.2 Performance of genus models on Regular. 110

Curriculum Vitae



Professional Experience

Osnabrück University · ***Research Assistant***
August 2014 – August 2020
Publication of several papers. Attendance at many international conferences/meetings. Strong contributions towards development of Open Graph Drawing Framework (OGDF). Management of student workforce. Supervision of theses.

Osnabrück University · ***Student Assistant***
February 2014 – August 2015
OGDF Development. Implementation of various graph algorithms, e.g., for Steiner Tree and Maximum Flow. Promoted to Research Assistant.

Freelance Web Developer
March 2008 – August 2015
Experienced in several content management systems, JavaScript frameworks, and Ruby on Rails. Terminated in favor of research position.

Education

Osnabrück University · ***M.Sc. Computer Science***
February 2014 – July 2015
Passed with distinction. Secondary subject: Systems Science. Planning

and development of ViOLa framework (Visualization and Optimization of Storage Strategies).
Master thesis: *Extracting proofs for crossings numbers.*

Osnabrück University · *B.Sc. Mathematics/Computer Science*
October 2009 – February 2014
Passed with distinction. Focused on computer science. Secondary subject: Systems Science.
Bachelor thesis: *A web interface for distributing high school courses.*

Refereed Conference Papers

2020

- Fritz Bökler, Markus Chimani, Mirko H. Wagner, and Tilo Wiedera. "An Experimental Study of ILP Formulations for the Longest Induced Path Problem." In: *Proceedings of the 6th International Symposium on Combinatorial Optimization (ISCO 2020), Montreal, QC, Canada.* Vol. 12176. LNCS. Springer, 2020, pp. 89–101. DOI: 10.1007/978-3-030-53262-8_8

- Alan Arroyo, Fabian Klute, Irene Parada, Raimund Seidel, Birgit Vogtenhuber, and Tilo Wiedera. "Inserting one edge into a simple drawing is hard." In: *Proceedings of the 46th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2020), Leeds, West Yorkshire, Great Britain.* LNCS. to appear. Springer, 2020

2019

- Drago Bokal, Zdeněk Dvořák, Petr Hliněný, Jesús Leaños, Bojan Mohar, and Tilo Wiedera. "Bounded Degree Conjecture Holds Precisely for c-Crossing-Critical Graphs with $c \leq 12$." In: *Proceedings of the 35th International Symposium on Computational Geometry (SoCG 2019), Portland, Oregon, USA*. vol. 129. LIPIcs. 2019, pp. 14:1–14:15. DOI: 10.4230/LIPIcs.SoCG.2019.14

- Markus Chimani and Tilo Wiedera. "Stronger ILPs for the Graph Genus Problem." In: *Proceedings of the 27th Annual European Symposium on Algorithms (ESA 2019), Munich/Garching, Germany.* Vol. 144. LIPIcs. 2019, pp. 30:1–30:15. DOI: 10.4230/LIPIcs.ESA.2019.30

2018

- Markus Chimani and Tilo Wiedera. "Cycles to the Rescue! Novel Constraints to Compute Maximum Planar Subgraphs Fast." In: *Proceedings of the 26th Annual European Symposium on Algorithms, (ESA 2018), Helsinki, Finland*. Vol. 112. LIPIcs, 2018, pp. 19:1–19:14. DOI: 10.4230/LIPIcs.ESA.2018.19

- Markus Chimani, Ivo Hedtke, and Tilo Wiedera. "Exact Algorithms for the Maximum Planar Subgraph Problem: New Models and Experiments." In: *Proceedings of the 17th International Symposium on Experimental Algorithms (SEA 2018), L'Aquila, Italy*. Vol. 103. LIPIcs. 2018, 22:1–22:15. DOI: 10.4230/LIPIcs.SEA.2018.22

2016

- Markus Chimani and Tilo Wiedera. "An ILP-based Proof System for the Crossing Number Problem." In: *Proceedings of the 24th Annual European Symposium on Algorithms (ESA 2016), Aarhus, Denmark*. Vol. 57. LIPIcs, 2016, pp. 29:1–29:13. DOI: 10.4230/LIPIcs.ESA.2016.29

- Markus Chimani, Karsten Klein, and Tilo Wiedera. "A Note on the Practicality of Maximal Planar Subgraph Algorithms." In: *Proceedings of the 24th International Symposium on Graph Drawing and Network Visualization, (GD 2016), Athens, Greece*. Vol. 9801. LNCS, 2016, pp. 357–364. DOI: 10.1007/978-3-319-50106-2_28

- Markus Chimani, Ivo Hedtke, and Tilo Wiedera. "Limits of Greedy Approximation Algorithms for the Maximum Planar Subgraph Problem." In: *Proceedings of the 27th International Workshop on Combinatorial Algorithms (IWOCA 2016), Helsinki, Finland*. Vol. 9843. LNCS, 2016, pp. 334–346. DOI: 10.1007/978-3-319-44543-4_26

Journal Articles

- Markus Chimani, Ivo Hedtke, and Tilo Wiedera. "Exact Algorithms for the Maximum Planar Subgraph Problem: New Models and Experiments." In: *ACM Journal of Experimental Algorithmics* 24.1 (2019). DOI: 10.1145/3320344

Articles Submitted for Review

- Markus Chimani, Niklas Troost, and Tilo Wiedera. *Approximating Multistage Matching Problems.* Submitted in 2020. arXiv: 2002.06887
- Drago Bokal, Markus Chimani, Alexander Nover, Jöran Schierbaum, Tobias Stolzmann, Mirko H. Wagner, and Tilo Wiedera. *Properties of Large 2-Crossing-Critical Graphs.* Submitted in 2020.
- Drago Bokal, Zdeněk Dvořák, Petr Hliněný, Jesús Leaños, Bojan Mohar, and Tilo Wiedera. *Bounded Degree Conjecture Holds Precisely for c-Crossing-Critical Graphs with* $c \leq 12$. Journal version submitted in 2020.

Reviewer for

- **ALENEX** Symposium on Algorithm Engineering and Experiments
- **Disc. Math.** Journal of Discrete Mathematics
- **ESA** European Symposium on Algorithms
- **GCOM** Journal of Graphs and Combinatorics
- **GD** Symposium on Graph Drawing and Network Visualization
- **JGAA** Journal of Graph Algorithms and Applications
- **SODA** Symposium on Discrete Algorithms