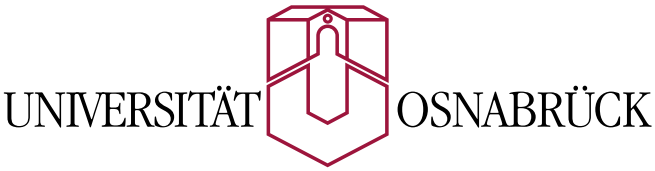


Minimum Genus and Maximum Planar Subgraph: Exact Algorithms and General Limits of Approximation Algorithms

Dissertation
zur Erlangung des Doktorgrades (Dr. rer. nat.)
des Fachbereichs Mathematik und Informatik
der Universität Osnabrück

vorgelegt von
Dipl.-Math. Ivo Hedtke

geboren am 30. September 1986 in Saalfeld/Saale

23. Juni 2017

Contents

Chapter 1

Introduction

This chapter collects the fundamental background for the following three chapters as well as an introduction to the problems we study in this thesis. Furthermore, an overview of the remaining text is given. Finally, we discuss four classes of test instances we used for our experimental evaluations.

Parts of this thesis were published in

- Stephan Beyer, Markus Chimani, Ivo Hedtke, and Michal Kotrbčík. "A Practical Method for the Minimum Genus of a Graph: Models and Experiments." In: *Experimental Algorithms - 15th International Symposium, SEA 2016, St. Petersburg, Russia, June 5-8, 2016, Proceedings*. Ed. by Andrew V. Goldberg and Alexander S. Kulikov. Vol. 9685. Lecture Notes in Computer Science. Springer, 2016, pp. 75–88. ISBN: 978-3-319-38850-2. DOI: 10.1007/978-3-319-38851-9_6

  Michal Kotrbčík presented the results at the 15th International Symposium on Experimental Algorithms (2016) in St. Petersburg.

- Markus Chimani, Ivo Hedtke, and Tilo Wiedera. "Limits of Greedy Approximation Algorithms for the Maximum Planar Subgraph Problem." In: *Combinatorial Algorithms - 27th International Workshop, IWOCA 2016, Helsinki, Finland, August 17-19, 2016, Proceedings*. Ed. by Veli Mäkinen, Simon J. Puglisi, and Leena Salmela. Vol. 9843. Lecture Notes in Computer Science. Springer, 2016, pp. 334–346. ISBN: 978-3-319-44542-7. DOI: 10.1007/978-3-319-44543-4_26

  Tilo Wiedera presented the results at the 27th International Workshop on Combinatorial Algorithms (2016) in Helsinki.

1.1 Mathematical Preliminaries

In this section we give fundamental definitions and review results we will use later in the text. The familiar reader can skip this section safely.

For convenience, we write $[k] := \mathbb{Z}_k = \{0, 1, \ldots, k-1\}$; addition and subtraction are modulo k. By $\uplus$ we denote the disjoint union of two sets.

1.1.1 Graph Theory

Graphs. Our terminology is standard and consistent with [MT01] and [KV12]. Unless otherwise stated we always consider undirected, simple, finite, connected graphs $G = (V, E)$ on finitely many vertices/nodes V and edges E.

An edge $e \in E$ between the vertices v and u is denoted as $e = vu = uv$. In this case v and u are called *incident* to e. Two vertices are *adjacent* if and only if there is an edge between them. Two adjacent vertices are called *neighbors*. The set of all neighbors of a vertex v is denoted by $N(v)$.

Multiple edges between the same pair of nodes are called *parallel*. An edge $e = vv$ is called a *(self-)loop* at v. Graphs without parallel edges and selfloops are called *simple*.

In a simple graph, the number of neighbors of a vertex v is called the *degree* of v and denoted by d_v. The maximum vertex degree in a graph G is denoted by $\Delta(G)$. A vertex of degree zero is called an *isolated vertex*. A graph whose vertices have the same degree d is called *d-regular*. A 3-regular graph is called *cubic*.

By $V(G)$ and $E(G)$ we denote the *vertex set* and *edge set* of a graph G, respectively. The *density* of a graph (V, E) is defined as $|E|/|V|$, the edges per node.

Let k be an arbitrary number. A *k-coloring* of a graph is map $c\colon V \to [k]$ such that for each edge $e = vw$ we have $c(v) \neq v(w)$. A graph is *k-colorable* if there exists a k-coloring for it.

Subgraphs. Let G and H be graphs. If $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$ we call H a *subgraph* of G. For a subset $S \subseteq V(G)$ we define the *induced subgraph* $G[S] = (S, E_S)$ by $E_S := \{vu \in E(G) : v, u \in S\}$. For a subset $S \subseteq E(G)$ we define the *induced subgraph* $G[S] = (V_S, S)$ by $V_S := \{u, v \in V : vu \in S\}$. A subgraph H of G with $V(G) = V(H)$ is called a *spanning* subgraph.

A *path* in G is a sequence of pairwise distinct vertices $v_1 v_2 \cdots v_\ell$ such that $v_i v_{i+1} \in E$ for all $i = 1, 2, \ldots, \ell - 1$. If we add the edge $v_\ell v_1$ to such a path we obtain a *cycle*. A k-path is a path of k edges. A *Hamiltonian path* (*Hamiltonian cycle*) is a path (cycle) that visits every vertex of the graph exactly once, respectively.

The length $|c|$ of a cycle c is the number of its edges. Let G be a graph and C_G be the set of the cycles of G. The *girth* of G is defined as $\min\{|c| : c \in C_G\}$.

The *complete graph* K_n is the graph on n vertices in which any two vertices are adjacent. The *complete bipartite graph* $K_{n,m}$ is the graph with vertices $\{v_0, \ldots, v_{n-1}\} \uplus \{u_0, \ldots, u_{m-1}\}$ and edges $\{v_i u_j : i \in [n], j \in [m]\}$. The graph K_n^- is defined as K_n where a single edge is deleted.

The *Wheel of n spokes* is the graph W_n obtained from a cycle C_n on n vertices by adding a new vertex and joining it to all vertices of C_n.

A triangle is a K_3 subgraph. A *cactus structure* is a graph whose cycles (if any) are triangles.

Operations on Graphs. The *Cartesian product* of two graphs G and H is the graph $G \square H$ with vertex set $V(G) \times V(H)$, in which two vertices (u, v) and (u', v') are adjacent if either $u = u'$ and $vv' \in E(H)$, or $v = v'$ and $uu' \in E(G)$. See Figure 2.A for an example of $K_3 \square K_3 \square K_3$.

For a subset $S \subseteq V(G)$ of the vertices of a graph G we write $G - S$ or $G \setminus S$ to denote the graph $G[V(G) \setminus S]$. If $S = \{s\}$ has only one element we also use the shorthand notation $G - s$ or $G \setminus s$.

For a subset $S \subseteq E(G)$ of the edges of a graph G we write $G - S$ or $G \setminus S$ to denote the graph obtained from G by deleting the edges in S. Again, we write $G - s$ or $G \setminus s$ if $S = \{s\}$.

Let u and v be two nodes of G. By $G + uv$ we denote the graph $\big(V(G), E(G) \cup \{uv\}\big)$.

Let G be a graph and $e = uv$ be one of its edges. By G/e we denote the *edge contraction* of e which is obtained from G by removing the edge e and identifying its ends u and v into a new vertex.

Connectivity. A graph is *connected* if there is a path between any two of its vertices. A *connected component* is a maximal connected subgraph.

A graph G is *k-connected* if it has at least $k+1$ vertices and for any subset $S \subseteq V(G)$ of at most $k-1$ vertices, $G - S$ is connected.

A *biconnected component* of a graph G is a maximal 2-connected subgraph of G.

Spanning Trees. A *tree* is a connected graph without cycles. A set of vertex-disjoint trees is called a *forest*. A spanning subgraph that is a tree is called a *spanning tree*.

Minors and Subdivisions. A graph G is a *minor* of a graph H if it can be obtained from H by a series of the following operations: delete a vertex, delete an edge, or contract an edge.

Given a graph G and one of its edges $e = vw$. We say that a graph H results from G by subdividing e if $V(H) = V(G) \uplus \{x\}$ and $E(H) = \{vx, xw\} \cup E(G) \setminus e$. A graph resulting from G by successively subdividing edges is called a *subdivision* of G.

A *Kuratowski subdivision* is a K_5 or $K_{3,3}$ subdivision.

Darts. The notation of "darts" is borrowed from [Fox+16].

Let E be the edge set of a graph G. The *dart set* $A(G)$ is defined as $E \times \{+1, -1\}$. For an edge $e \in E$, the darts of e are $(e, +1)$ or e^+ and $(e, -1)$ or e^-, respectively. We think of the darts of e as oriented versions of e, one for each orientation. The *reversed dart* $\mathrm{rev}((e, i))$ is defined as $(e, -i)$. Darts are also known as "directed edges" and "halfedges". We use the notation $u \to v$ to denote a dart with orientation from u to v. When the context is clear, we sometimes simply write uv.

Let v be a vertex of a graph. By $\delta^+(v)$ we denote the set of out-going darts, *i.e.*, the set $\{vx \in A\}$. By $\delta^-(v)$ we denote the set of in-going darts, *i.e.*, the set $\{vx \in A\}$.

Non-Planar Core. The non-planar core reduction by Chimani and Gutwenger (see [CG09]) reduces a given graph to a core where each maximal planar 2-component is replaced by a (weighted) edge. We will use this core as input for our algorithms whenever the problem is susceptible to such a reduction. Here we give a formal definition and cite some known results. The presentation in this paragraph is based on [CG09].

Definition 1.1 (planar 2-component). Let $s, t \in V$ be two distinct vertices of a 2-connected, non-planar graph. An edge-induced subgraph $C = G[E_C]$ is called a *planar 2-component* (or *planar st-component*) of G if $C + st$ is planar and $V(C) \cap V(G[E \setminus E_C]) = \{s, t\}$. A single edge is called a *trivial planar 2-component*. 1.1

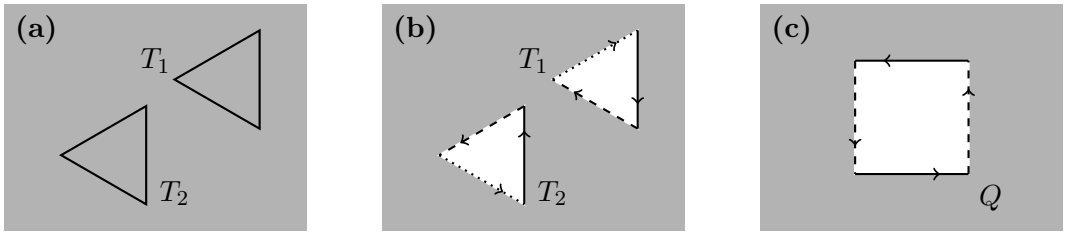
A non-trivial planar 2-component C of G is called a *maximal planar 2-component* of G if and only if there is no planar 2-component C' of G with $C \subsetneq C'$. ⌟

Definition (non-planar core). The *(weighted) non-planar core* $(\mathcal{C}, w)$ of G is a graph $\mathcal{C}$ with a weight function $w\colon E \to \mathbb{N}$ such that $\mathcal{C}$ is a copy of G in which each maximal planar st-component C of G is substituted with a virtual edge $e_C = st$ with weight $w(e_C) = \mathrm{mincut}_{s,t}(C)$, and each non-virtual edge e has weight $w(e) = 1$. ⌟

1.1.2 Group Theory

Cayley graphs. We follow [Jun13, Sect. 9.4] in his definition of Cayley graphs. He also shows some interesting results in the field of Cayley graphs. We use them as an important class of test instances in the next chapter. See Section 2.1 for an example.

Definition (Cayley graph). Let H be a finite group and $S \subsetneq H$ be a subset of H with the properties that $1 \notin S$ and $S = S^{-1} := \{s^{-1} : s \in S\}$. The *Cayley graph* $\mathrm{CG}(H, S) = (H, E)$ of H with respect to S has H as its vertices and the edges $E = \{xy : xy^{-1} \in S\}$. ⌟



1.A **Figure 1.A: (a)** A surface with two triangles T_1 and T_2. **(b)** A handle that is created by deleting the interiors of T_1 and T_2 and identifying the stroked, dashed and dotted darts each. **(c)** A crosscap that is created by deleting the interior of a quadrangle Q and identifying the stroked and dashed darts each.

Definition (generating subset). A *generating subset* of a group is subset such that every element of the group can be expressed as the combination of the elements of the subset and their inverses. ⌟

An alternative, more practical definition is given by Cayley himself in [Cay78]: Let H be a finite group and Ω be a generating subset. The Cayley graph is a directed graph $\mathrm{CG}(H) = (H, A)$ with H as its vertices and the arcs $A = \bigcup_{\omega\in\Omega} A_\omega$ where each $A_\omega = \{h \to h\omega : h \in H\}$ is colored in a different color c_ω. The colors in Figure 2.A are induced by the three generators of $\mathbb{Z}_3^3 \cong \mathrm{Sym}((1,2,3),(4,5,6),(7,8,9))$.

1.1.3 Surfaces, Embeddings, Facial Walks and the Face Traversal Procedure

Surfaces. We give a compressed version of the definitions in [MT01, Section 3.1].

A *surface* is a connected compact Hausdorff topological space which is locally homeomorphic to an open disc in the plane. We do not give a precise definition, the interested reader is referred to [MT01, Section 3.1].

By $\mathbb{S}_0$ we denote the sphere. It has *orientable* and *non-orientable genus* zero. If we add h handles to $\mathbb{S}_0$, we obtain the surface $\mathbb{S}_h$ which we refer to as the *orientable surface of genus h*. If we add h crosscaps to $\mathbb{S}_0$, we get the *non-orientable surface* $\mathbb{N}_h$ *of genus h*. We continue with the definition of handles and crosscaps.

Consider the drawing of two disjoint triangles T_1 and T_2 such that all six sides have the same length) on a surface S. We form a new surface S' by deleting the interior of T_1 and T_2 and identifying T_1 with T_2 such that the clockwise orientations around T_1 and T_2 disagree, c.f. Figure 1.A(b). We say that the surface S' is obtained from S by adding a *handle*.

Let Q be a quadrangle with equilateral sides on a surface S. Let S' be the surface obtained by deleting the interior of Q and identifying diametrically opposite points of the quadrangle as shown in Figure 1.A(c). Then S' is said to be obtained from S by adding a *crosscap*.

Lemma. [MT01, Theorem 3.1.3] *Every surface is homeomorphic to precisely one of the surfaces* $\mathbb{S}_h$ *(*$h \geq 0$*) or* $\mathbb{N}_k$ *(*$k > 0$*).*

Embeddings. We give a short description of 2-cell embeddings based on [MT01, Section 3.1] but our focus is on combinatorial embeddings [MT01, Section 4.1].

A *polygon* is a plane figure that is bounded by a finite chain of straight line segments closing in a loop. Let F be a finite set of pairwise disjoint polygons in the plane with all *sides* (segments) of unit length. Suppose that all polygons together have m sides $\sigma_1, \ldots, \sigma_m$, where m is even. Orient arbitrarily each of the sides by choosing one of its endpoints as the initial point, and choose an arbitrary partition of the sides into pairs. From the disjoint union of polygons in F we form a topological space S by identifying sides in given pairs of the partition in such a way

that the orientations agree. We get a compact Hausdorff space S which is locally homeomorphic to the unit disc in the plane. The sides $\sigma_1, \ldots, \sigma_m$ and their endpoints determine a connected multigraph G embedded in S. We say that G is *2-cell embedded* in S. The polygons in F are the *faces* of G. If all faces are triangles (K_3) and G is a graph, we say that G *triangulates* the surfaces, and G is *triangulated*.

The *Euler characteristic* $\chi(S)$ of a surface S is defined as

$$\chi(S) := \begin{cases} 2-2h & S = \mathbb{S}_h, \\ 2-k & S = \mathbb{N}_k. \end{cases}$$

Lemma (Euler's formula). *Let G be a graph which is 2-cell embedded in a surface S. If G has n vertices, m edges and f faces in S, then $n - m + f = \chi(S)$.*

An *embedding scheme* of a connected graph G is a pair $\Pi = (\lambda, \pi)$ where $\pi = \{\pi_v : v \in V(G)\}$ is a *rotation system* (*i.e.*, for each vertex v, π_v is a cyclic permutation of $N(v)$) and $\lambda\colon E(G) \to \{+1, -1\}$ is a signature mapping which assigns each edge $e \in E(G)$ a *sign* $\lambda(e)$. If an edge e is incident with v, then the cyclic sequence $e, \pi_v(e), \pi_v(\pi_v(e)), \ldots$ is called the Π-*counter-clockwise ordering around* v.

The Face Traversal Procedure. The procedure defined in this paragraph is fundamental for our work. We will build exact algorithms that simulate this procedure. We also base some definitions on it.

Definition (Π-facial walks, face traversal procedure). [MT01, p. 93] Let $\Pi = (\lambda, \pi)$ be an embedding scheme for a given embedding of a graph G. A Π-*facial walk* is a closed walk on the dart set of G, defined by the *face traversal procedure*: By $\sigma \in \{-1, 1\}$ we denote the current state of the walk and initialize it with $\sigma := 1$. We start with an arbitrary vertex v and an edge $e = vu$ incident with v. Traverse the edge e from v to u. We are now in u and update the current state as $\sigma := \lambda(e)\sigma$. We continue the walk along $\pi_u^\sigma(e)$. The walk is completed when the initial edge e is encountered in the same direction from v to u with $\sigma = 1$. ⌟

Definition (positive signature). The signature λ of an embedding scheme (λ, π) for a graph $G = (V, E)$ is called *positive* if and only if $\lambda(e) = 1$ for all $e \in E$. We write $\lambda \equiv 1$ in short. ⌟

Lemma 1.2. [MT01, Lem. 4.1.4] *An embedding is orientable if and only if it is equivalent to an embedding with positive signature.* 1.2

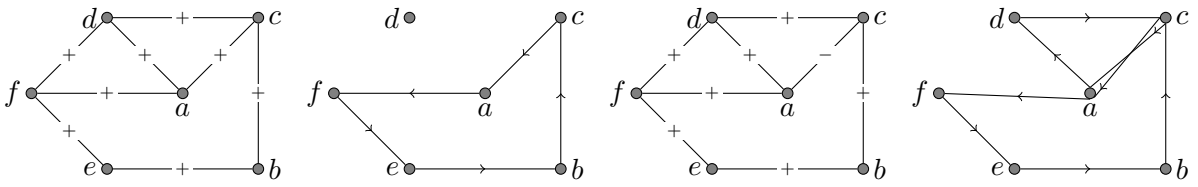
For the orientable case the procedure above can be simplified by $\lambda \equiv 1$. An example can be found in Figure 1.B.

Lemma 1.3. *Let $e = uv$ be an edge, $S_{\mathrm{O}} := A(e) = \{uv, vu\}$, and $S_{\mathrm{N}} := A(e) \times \{-1, +1\} = \{(uv, -1), (uv, +1), (vu, -1), (vu, +1)\}$.* 1.3

a) Let $G = (V, E)$ be a graph embedded on an orientable surface. For each edge $e \in E$ the set of all facial walks uses exactly two elements of $S_{\mathrm{O}}(e)$.

b) Let $G = (V, E)$ be a graph embedded on a non-orientable surface. For each edge $e \in E$ the set of all facial walks uses exactly two elements of $S_{\mathrm{N}}(e)$.

Proof. See the proof of Theorem 1 in [BD09]. □



1.B **Figure 1.B:** A graph with one rotation system but two different signatures and the induced sets of facial walks. A sample facial walk in the left graph with $\lambda \equiv 1$ is $(f,+) \xrightarrow{+} (e,+) \xrightarrow{+} (b,+) \xrightarrow{+} (c,+) \xrightarrow{+} (a,+) \xrightarrow{+} (f,+)$, where (v,σ) denotes the current vertex v and the current state σ of the walk. In the right graph the signature of $\{b,c\}$ is -1. A sample facial walk in the right graph starting the same edge as before is $(f,+) \xrightarrow{+} (e,+) \xrightarrow{+} (b,+) \xrightarrow{+} (c,+) \xrightarrow{-} (a,-) \xrightarrow{+} (d,-) \xrightarrow{+} (c,-) \xrightarrow{-} (a,+) \xrightarrow{+} (f,+)$.

Definition (Euler genus). The *Euler genus* $\mathrm{eg}(\Pi)$ of an embedding scheme $\Pi = (\lambda, \pi)$ is defined as $\mathrm{eg}(\Pi) := 2 - \chi(\Pi)$. ⌟

A graph is called a *plane graph* if an embedding of it in $\mathbb{S}_0$ is given. A graph is called *planar* if an embedding in $\mathbb{S}_0$ exists. A *planar embedding* of a planar graph is an embedding in $\mathbb{S}_0$ together with one face marked as the *outer face* or *infinite face* f_∞: Consider an embedding of a planar graph in $\mathbb{S}_0$, which means on a sphere. For each ϵ we can transform the embedding in such a way that the graph is located inside a disc of radius ϵ. This disc is homeomorphic to a plane. Thus, we have an embedding of the graph in the plane where we now have one unbounded face, the outer face f_∞. A graph is called *outerplanar* if it is planar and there is a face that visits each vertex of the graph at least once.

For a given face in an embedding we also call the according facial walk the *boundary* of the face. The facial walk of a face is a directed cycle which separates two regions from each other. Thus, we can speak of the *interior* and exterior of a face.

1.1.4 Linear Programming and Satisfiability Formulations

Linear Program. We follow Korte and Vygen [KV12] for the basics on linear programming.

A *polyhedron* in $\mathbb{R}^n$ is a set $\{x \in \mathbb{R}^n : Ax \leq b\}$ for some matrix $A \in \mathbb{R}^{m \times n}$ and some vector $b \in \mathbb{R}^m$. A bounded polyhedron is called a *polytope*.

Let P be a non-empty polyhedron. If c is a vector for which $\delta := \{c^\top x : x \in P\}$ is finite, then $\{x : c^\top x = \delta\}$ is called a *supporting hyperplane* of P. A *face* of P is P itself or the intersection of P with a supporting hyperplane of P.

By $c^\top x$ we denote the scalar product of two vectors c and x. The Linear Programming Problem reads as follows:

> Linear Programming
>
> **Instance:** A polyhedron P in $\mathbb{R}^n$ and a vector $c \in \mathbb{R}^n$.
> **Task:** Find a element $x \in P$ such that $c^\top x$ is maximum, decide that P is empty, or decide that for all $\alpha \in \mathbb{R}$ there is an $y \in P$ such that $c^\top y > \alpha$.

A *linear program (LP)* is a instance of the Linear Programming Problem. A *feasible solution* of an LP is a vector $x \in P$. A feasible solution attaining the maximum value is called *optimum solution*.

There are two possibilities when an LP has no solution: The problem is either *infeasible* (*i.e.*, $P = \emptyset$) or *unbounded*. If an LP is neither infeasible nor unbounded it has an optimum solution.

We will write LPs as

$$\begin{aligned} \max \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \mathbb{R}^n \end{aligned}$$

A *constraint* is induced by a row $a^{(i)}$ of the matrix A, it is the inequality $a^{(i)}x \leq b_i$.

In most cases we consider *integer linear programs (ILP)* where we work over $\mathbb{Z}$: $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{Z}^m$, $c, x \in \mathbb{Z}^n$. The LP obtained from an ILP by switching back to $\mathbb{R}$ is called *LP-relaxation*.

Separation. An LP can be solved in polynomial time, for example by the so-called *Ellipsoid Method*. But this requires that the polyhedron is given explicitly by a list of inequalities. But an LP can be solved in polynomial time (independent of the number of constraints) too, if we have a so-called *separation oracle* (with polynomial run-time): a method for the following problem:

> Separation Problem
>
> **Instance:** A convex set $P \subseteq \mathbb{R}^n$ and a vector $y \in \mathbb{Q}^n$.
> **Task:** Either decide that $y \in P$ or find a vector $d \in \mathbb{Q}^n$ and a scalar $d_0 \in \mathbb{Q}$ such that $d^\top x \leq d_0 < d^\top y$ for all $x \in P$.

The set $\{x : d^\top x = d_0\}$ is called a *cutting plane*.

We will use cuttings planes to solve ILPs by successively solving LP-relaxations and cut away parts of the polyhedron (by adding new constraints) in hope of obtaining an integer solution. The Separation problem searches for a cutting plane. Any cutting plane found is added to the LP and the LP is solved again.

Branch-and-Bound (B&B). We compute cutting planes for two reasons:

1. A class of constraints in our ILP formulation is too large (*i.e.*, exponentially large).
2. The inequalities of the LP-relaxation are not sufficient to yield an integer solution. We search for valid inequalities (that do not cut integer solutions in the polyhedron) that cut of those fractional solutions.

For example if we have too many constraints of a certain class we omit them in a relaxed formulation and use cutting planes by checking if solutions of the relaxed formulation violate constraints of our class. We repeat the cutting step until an integer solution is found that satisfies all constraints, or until we cannot find a violated inequality. In this case, we use a branch step.

Consider the case where we deal with a 0/1-ILP, *i.e.*, an ILP where we substituted the $x \in \mathbb{Z}^n$ constraint by $x \in \{0, 1\}^n$. In a branch step we now choose a variable x_i and consider the two sub-cases $x_i = 0$ and $x_i = 1$. Repeated application produces a enumeration tree of possible cases. The enumeration tree of all possible variable settings is partially traversed, computing local upper bounds and global lower bounds, which are used to cut off parts of the tree that cannot produce the optimum value.

An upper bound (optimum value of the LP-relaxation) for a sub-problem in the enumeration tree is called a *local upper bound*. If the solution of a sub-problem is feasible for the original problem and is an integer solution, its value becomes a *global lower bound*. Subsequently, we cut off sub-problems with a local upper bound not greater than the best global upper bound.

The ILP solvers we will use are based on the Branch & Bound method to find integer solution from LP-relaxations. We will also extend the B&B process with self-constructed separation oracles when we have a constraint class that is not polynomially bounded.

SAT and PBS Formulations. A Boolean formulae uses variables, the constants `true`, `false`, unary, and binary operators. The language of the propositional calculus is recursively defined by

$$x_1 \mid \cdots \mid x_k \mid \texttt{true} \mid \texttt{false} \mid \neg\phi \mid (\phi \vee \psi) \mid (\phi \wedge \psi)$$

where $x_1, \ldots, x_k$ are the variables and ψ and ϕ are propositional formulae. Furthermore, we define $\phi \to \psi := \neg\phi \vee \psi$ and $\phi \leftrightarrow \psi := (\phi \to \psi) \wedge (\psi \to \phi)$.

An *assignment* is a map $\{x_1, \ldots, x_k\} \to \{\texttt{true}, \texttt{false}\}$. An assignment α is a *satisfying assignment* if the substitution of x_i by $\alpha(x_i)$ in the underlying propositional formulae evaluates to `true`.

Definition (conjunctive normal form (CNF)). Let V be a set of variables. A literal ℓ is a variable v or the negation $\neg v$ of a variable. A propositional formulae ϕ is in conjunctive normal form (CNF) if

$$\phi = \bigwedge\nolimits_{i \in I} \bigvee\nolimits_{j \in I_i} \ell_{i,j}$$

where $\ell_{i,j}$ are literals and I and I_i (for $i \in I$) are finite index sets. ⌟

The definition above allows us to say that a CNF is a conjunction of disjunctions, or a conjunction of *clauses*. We call this a *SAT formulation*.

The Satisfiability Problem is:

> Satisfiability (SAT)
>
> **Instance:** A propositional formulae ϕ in CNF.
> **Task:** Is there a satisfying assignment for ϕ?

Most of the problems we consider in this work are optimization problems such as the Linear Programming problem. A SAT formulation can be used to compute the optimum of a bounded ILP by iteratively answering the question "Is there a feasible solution x such that $c^\top x \geq d$?" for given values of d. An alternative is to use a *Pseudo-Boolean Satisfiability (PBS)* formulation. Each clause (or *constraint*) c_i has the form

$$\sum\nolimits_{j=1}^{k} a_{i,j}\ell_j \geq b$$

where ℓ_i is a literal and $a_{i,j}, b$ are coefficients in $\mathbb{Z}$. If we have an assignment, we interpret the literals ℓ_i as 1 or 0 if the literal evaluates to `true` or `false`, respectively.

A *PBS formulation* consists of a set of constraints together with an objective function which in this case is a linear combination of literals with coefficients in $\mathbb{Z}$.

We define the generalization of SAT as

> Pseudo-Boolean Satisfiability (PBS)
>
> **Instance:** A PBS formulation with constraints C and an objective function z.
> **Task:** Find a satisfying assignment α for C that maximizes z.

1.1.5 Approximation Algorithms

As there is no hope to find a polynomial-time algorithm for an NP-hard problem, unless $P = NP$, we consider the important concept of approximation algorithms.

Definition (approximation algorithm). Let $\mathcal{P}$ be an optimization (maximization) problem and $OPT(J)$ denote the value of an optimal solution for an instance J of $\mathcal{P}$. An *α-factor approximation algorithm* for $\mathcal{P}$ is a polynomial-time algorithm $\mathcal{A}$ such that

$$\alpha \cdot OPT(J) \leq A(J)$$

for all instances J of $\mathcal{P}$, where $A(J)$ denotes the solution value of algorithm $\mathcal{A}$ for instance J. We call α the *approximation ratio*, *approximation factor*, or *approximation guarantee*. ⌟

Example. From Euler's formula we know that a planar graph on n vertices has at most $3n-6$ edges. Any spanning tree for a connected graph on n vertices has $n-1$ edges. Thus, an algorithm that computes a spanning tree is a $\frac{1}{3}$-approximation algorithm for the Maximum Planar Subgraph problem (see next section), as

$$\frac{1}{3} \cdot (3n-6) = n-2 \leq n-1.$$ ⌟

1.2 Motivation

This thesis considers two measures for the non-planarity of a graph. A graph is planar if and only if it can be embedded in $\mathbb{S}_0$. A graph is either planar or non-planar. This classification can be extended by asking *how far away is a graph from planarity.* A variety of measures for non-planarity have been proposed. An overview can be found in [Lie01; Sch14]. Some prominent measures are

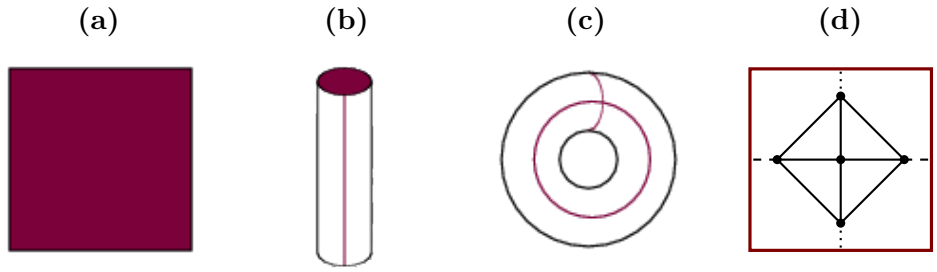
- *Crossing Number*: What is the minimum number of edge-crossings in any drawing in $\mathbb{S}_0$?
- *Skewness*: What is the minimum number of edges that have to be deleted from the graph to obtain a planar graph?
- *Thickness*: What is the minimum number of planar subgraphs whose union is the input graph itself?
- *Minimum Genus*: What is the minimum number g such that the graph can be embedded in $\mathbb{S}_g$.

All of the above problems are NP-hard, see [GJ83; LG79; Man83; Tho89]. We focus on skewness and minimum genus in this thesis. We will present exact algorithms for both measures as well as results on limits of approximation algorithms for the Maximum Planar Subgraph problem.

Minimum Genus (MGP)

Instance: Undirected, non-planar graph G.
Task: Find the minimum g such that G can be embedded in $\mathbb{S}_g$.

We denote the minimum genus of a graph G by $\gamma(G)$. We propose the first ILP and SAT formulations for the MGP. These formulations allow us to develop the first working implementations of general algorithms for the problem, other than exhaustive search. Generally, such modeling approaches are known for several planarity concepts and non-planarity measures (e.g., crossing number [Chi08; Chi09], skewness [JM96], upward planarity [CZ15]) and often attain surprisingly strong results. However, for the MGP it is at first rather unclear how to



1.C **Figure 1.C:** An embedding of the K_5 on the torus. (a) A piece of paper with one side white and one side red. (b) Transformation of the paper into the surface of the cylinder by identifying the vertical page borders with each other. (c) Transformation of the cylindrical surface into a torus by identifying the circular borders with each other. (d) A drawing of the K_5 on the white side of the paper. By transforming the paper as discussed before and identifying the dashed and the dotted edges with each other, we obtain an embedding of K_5 on $\mathbb{S}_1$.

capture the topological nature of the question in simple variables. To the best of our knowledge, there are no known formulations for this problem.

We also extend our formulation to attack the MGP on non-orientable surfaces.

Minimum Non-Orientable Genus

Instance: Undirected, non-planar graph G.
Task: Find the minimum h such that G can be embedded in $\mathbb{N}_h$.

Example. Consider the complete graph K_5 on five vertices. It is non-planar but can embedded on $\mathbb{S}_1$ (the torus). As it cannot have genus zero and there is an embedding on the torus, the minimum genus of K_5 is one. In fact, the minimum genus of the complete graph and complete bipartite graph is known since 1968, see [Rin55; Rin65a].

$$\gamma(K_n) = \left\lceil \frac{(n-3)(n-4)}{12} \right\rceil \qquad \text{and} \qquad \gamma(K_{m,n}) = \left\lceil \frac{(m-2)(n-2)}{4} \right\rceil$$

An embedding of the K_5 on the torus is shown in figures 1.C and 2.J (page 33). ⌟

Note that every rotation system of a graph $G = (V, E)$ corresponds to an embedding of the graph in a surface of some genus, say ℓ. As a surface of genus ℓ is homeomorphic to a sphere with ℓ added handles, it is always possible to embed G in $\mathbb{S}_{|E|}$ by using a handle for each edge of the graph.

In Chapter 2 we investigate exact algorithms that construct a rotation system which induces a minimum genus embedding.

The second non-planarity measure we discuss is the size of a maximum planar subgraph.

Maximum Planar Subgraph (MPS)

Instance: Undirected graph G.
Task: Find a planar subgraph of G with maximum edge cardinality.

This problem is an equivalent of the skewness problem: If we denote the skewness of a graph $G = (V, E)$ by $\mu(G)$ and the size of a maximum planar subgraph by $mps(G)$ it follows that

$$|E| = \mu(G) + mps(G).$$

It is known to be MaxSNP-hard [Căl+98, Theorem 4.1] and the currently best known approximation algorithm achieves a ratio of 4/9. The best practical approximation algorithm so far achieved an approximation ratio of 7/18. Recently, Chalermsook and Schmid [CS17] found a greedy algorithm that searches for structures denser than triangles. Their algorithm is the now best greedy approximation algorithm for MPS but it is unknown if it achieve the 4/9-ratio or better.

An application of MPS is discussed in Tamassia et al. [TDB88]. *Graph Drawing* is concerned with the geometric representation of graphs. It is motivated by applications where it is crucial to visualize structural information as graphs. There are no absolute criteria for *nice* graph drawings. Some example for commonly used properties are: minimizing the number of edge crossings or placing the vertices on a grid. A first step in a drawing algorithm that tries to minimizing the number of crossings typically is to start with a maximum planar subgraph of the input graph and then insert the remaining edges as "smart" as possible. Tamassia et al. used this approach for the automated drawing of entity relationship diagrams on a grid.

Exact algorithms for MPS are typically based on the extraction of Kuratowski subdivisions to add additional constraints in a cutting plane approach.

Theorem. [Kur30] *A graph is planar if and only if it does not contain a K_5- or $K_{3,3}$-subdivision.*

Kuratowski subdivisions can be found by linear-time planarity tests as [BM04]. It is also possible to efficiently extract multiple Kuratowski subdivision at once [CMS07].

In most cases we consider a generalized variant of the Maximum Planar Subgraph problem:

> Maximum Weighted Planar Subgraph
>
> **Instance:** Undirected graph $G = (V, E)$ with weights $w\colon E \to \mathbb{N}$.
> **Task:** Find a planar subgraph $H = (V_H, E_H)$ of G with maximum weight $w(E_H)$.

When developing algorithms, we also use the name Maximum Planar Subgraph for the problem above, as MPS is contained in it (by using unit weights) and our algorithms are for the weighted case.

We can extend the non-planarity measure skewness also for weighted graphs (in the same way as we defined the generalized problem above). We write $\mu(G, w)$ in the case of a graph G with weights w. The following result will be used later in the preprocessing step of algorithms for the MPS problem:

Lemma 1.4. [CG09, Theorem 14] *Let G be a 2-connected graph and $(\mathcal{C}, w)$ be its non-planar core. Then $\mu(G) = \mu(\mathcal{C}, w)$.* 1.4

1.3 Overview

This thesis is structured as follows:

2. Chapter *Exact Algorithms for the Minimum Genus Problem*: We present the first ILP and SAT formulations for the MGP. We investigate different ways to speed-up our algorithms in practice:
   - symmetry breaking constraints,
   - specialized variables for low-degree vertices,
   - binary face index representation,

- face skipping, and
- incremental formulations;

as well as polynomially sized formulations:

- index reformulation and
- betweenness reformulation.

Using the *Rome* and *North* graph instances we evaluate our approaches experimentally based on state-of-the-art ILP and SAT solvers. After comparing our 48 variants to each other on a small subset of all instances we compare our SAT formulation to the ILP formulation by Stephan Beyer. Furthermore, we discuss the performance of our approach against existing genus computations.

The developed algorithm works well on small to medium-sized graphs with small genus, and compares favorably to other approaches.

In addition to the original publication [Bey+16], we cover the new *face skipping* and *incremental formulation* speed-ups as well as the extension to the non-orientable case.

3. Chapter *Limits of Greedy Approximation Algorithms for the Maximum Planar Subgraph Problem*: Based on our attempts to find a new approximation algorithm for the MPS problem, we studied limits of greedy approximation algorithms in general.

   We generalized many results that held only for specific algorithms to classes of greedy algorithms that are natural extensions or variants of existing ones. Chapter 3 covers algorithms inspired by famous planarity test algorithms, and generalizations of algorithms building triangular structures.

   Furthermore, we present much shorter proofs for the facts that *maximal planar subgraph* yields a 1/3 approximation for MPS and for the NP-hardness of MPS itself.

   In addition to our publication [CHW16] we discuss the application of graph decompositions (cut- and path-width) to approximation algorithms. As we will see, we do not gain new results in this area but hope to highlight insights induced by specialized decompositions compared to the steamroller method of extended monadic second order logic for graphs with bounded tree-width. Finally, we show an alternative proof for the approximation ratio of the well known 7/18 approximation algorithm by Călinescu et al.

4. Chapter *Exact Algorithms for the Maximum Planar Subgraph Problem*: Besides the K_5- and $K_{3,3}$-subdivision criterion by Kuratowski there is a variety of planarity criteria. We give a non-complete list of 14 criteria where we picked four to build exact algorithms on them for the MPS problem:

   - *Additional Minors*: If we delete an edge from every Kuratowski subdivision while taking care of deleting as little as possible edges we obtain an MPS. The same principle can be used for graphs that are *not apex* (there is no edge such that its deletion results in a planar graph) where we have to delete at least two edges for every found subgraph. We obtain additional constraints that could strengthen the Kuratowski-based formulation.
   - *Total Orders*: A graph is planar if and only if there are three total orders $<_i$ on its vertices such that their intersection is empty and for each edge xy and each vertex $z \notin \{x, y\}$ there is an order such that $x <_i z$ and $y <_i z$. This criterion is related to Schnyder-Layout and admits straightforward ILP and PBS formulations.

- *Theta Graphs*: If there are three node-disjoint paths between a pair of claws in a graph the embedding of one claw induces the embedding of the other. The two claws together with the three paths form a *theta graph*. A graph is planar if and only if the embeddings of all claws does not contain a contradiction: Enforcing two different embeddings of one claw that if both realized result in a non-planar embedding.
- *Facial Walks*: A maximum planar subgraph corresponds to a maximum subgraph of genus zero. We use the insights gained in Chapter 2 on the MGP to develop exact algorithms that simulate the face traversal procedure on subgraphs to find the MPS. The formulations are more intricate than for the MGP as we are nor longer restricted to biconnected graphs with minimum vertex degree three. We examine the same speed-up and reformulation approaches as in Chapter 2 to optimize the performance of our algorithms.

Unfortunately, we were not able to beat the formulation based on Kuratowski subdivisions.

1.4 Test Instances and Used Frameworks

In this section we give a short overview of the used instance sets for our experiments and their characteristica.

Rome graphs. Based on 112 real life graphs from software companies, textbooks in software engineering and from various theses, Di Battista et al. [Di +97] generated 11 582 test graphs as variants of the real life instances. See [Di +97, Section 3.2] for details. We consider the 8 249 non-planar graphs from this instance set. The characteristica of the non-planar Rome graphs are shown in figures 1.D and 1.E.

North graphs. Based on the collection of 5 114 directed graphs obtained with an online graph drawing service by Stephen North, Di Battista et al. [Di +00] filtered the instance set to 1 277 instances. See [Di +00, Section 3.2] for details. We consider the 423 non-planar graphs. Their characteristica are shown in figures 1.D and 1.E

SteinLib. From the SteinLib instance library by Koch et al. [KMV00] we use the non-planar non-complete graphs in the classes

- B: Sparse graphs on 50–100 nodes.
- i080: Sparse graphs on 100 nodes.
- The single instances `cc6-2u.stp` ($|V| = 64$, $|E| = 192$), `cc6-2p.stp` (64, 192), `cc3-4p.stp` (64, 288), `cc3-4u.stp` (64, 288), `design432.stp` (8, 20), `hc6p.stp` (64, 192), and `hc6u.stp` (64, 192).

We ignore the weights and interpret the instances as undirected graphs.

Expander graphs. We use the generator [SW99] for regular graphs (implemented together with Tilo Wiedera as part of the OGDF) to create a set of instances that "seem to be the hardest instances" for MPS heuristics[CKW16]; they are expander graphs with high probability [Kow11, Section 3.5]. Expander graphs have the properties that they are fairly sparse albeit highly connected.

We generated 20 graphs for each configuration:

nodes	degree of each node
10	4, 6
20	4, 6, 10
30	4, 6, 10, 20
50	4, 6, 10, 20, 40
100	4, 6, 10, 20, 40
1 000	4, 6, 10, 20, 40
10 000	4, 6, 10, 20, 40

We complete this section with an overview of the used software and frameworks.

OGDF. All our implementations are based on the Open Graph Drawing Framework (OGDF) [Chi+13], a self-contained C++ class library for the automatic layout of diagrams.

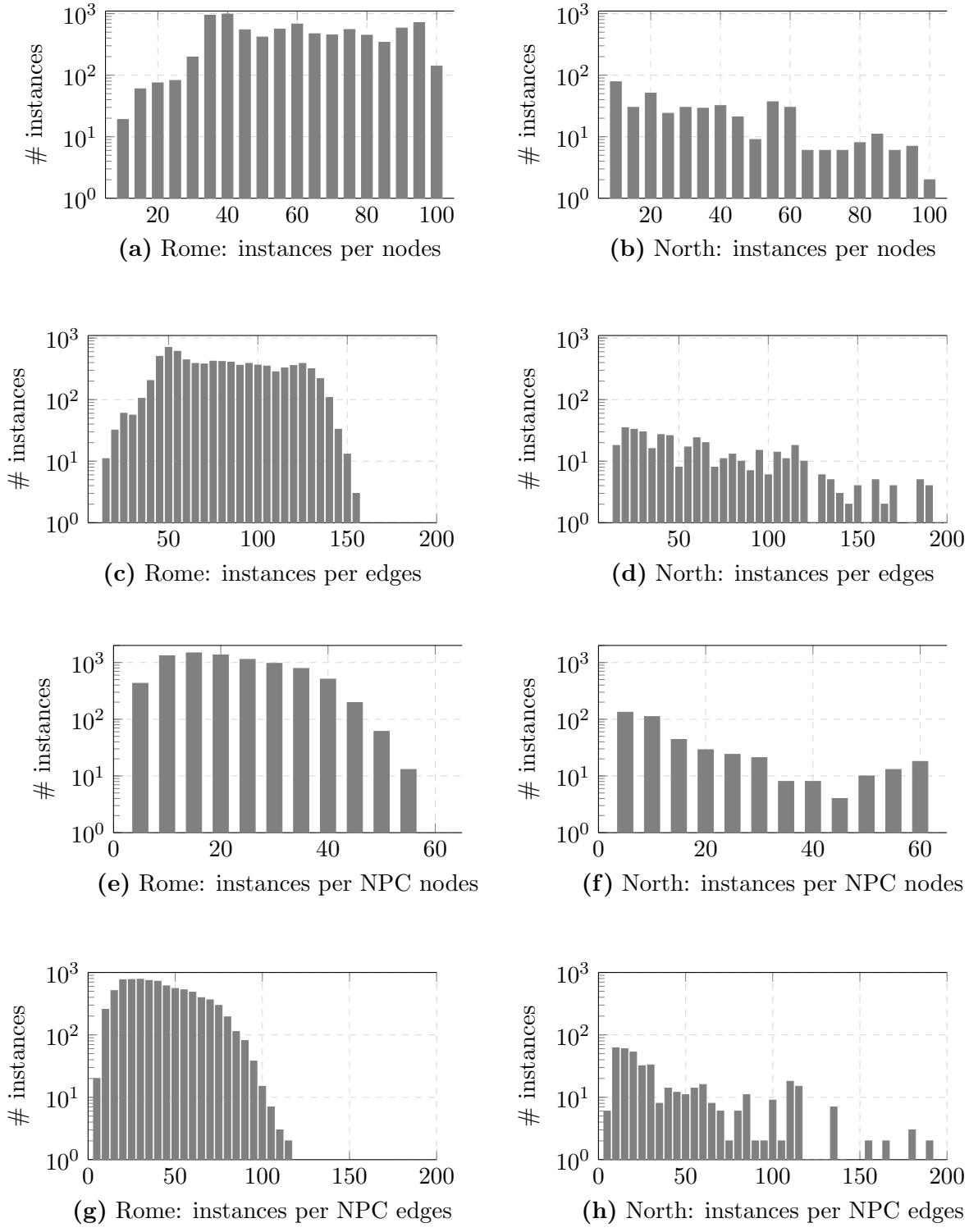
SCIP. The cutting plane methods in Chapter 4 are implemented as plug-ins for SCIP (Solving Constraint Integer Programs) [Mah+17].

Gurobi. We use Gurobi as LP solver in Chapter 4 [Gur16]. It is invoked by SCIP.

CPLEX. We use CPLEX as ILP-solver in Chapter 2.

Lingeling. The SAT formulations in Chapter 2 are solved by lingeling [Bie14].

Clasp. The PBS formulations in Chapter 4 are solved by Clasp [Geb+11].



(a) Rome: instances per nodes

(b) North: instances per nodes

(c) Rome: instances per edges

(d) North: instances per edges

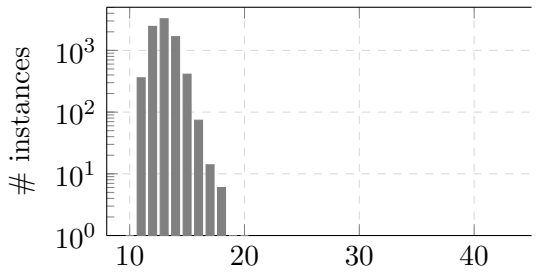
(e) Rome: instances per NPC nodes

(f) North: instances per NPC nodes

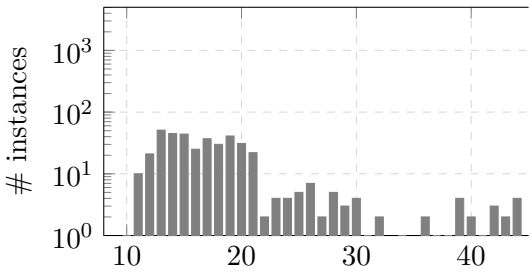
(g) Rome: instances per NPC edges

(h) North: instances per NPC edges

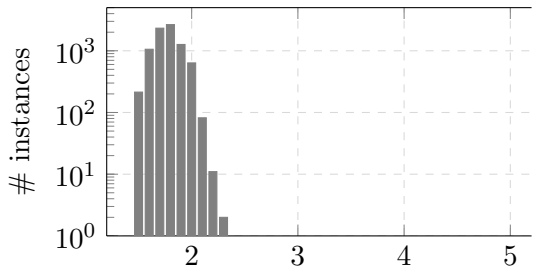
Figure 1.D: Characteristica *instances per given number of nodes* (grouped by $|V| \in \{5k, 5k+1, \ldots, 5k+4\}$), *instances per given number of edges* (grouped by $|E| \in \{5k, 5k+1, \ldots, 5k+4\}$), *instances per given number of nodes of the NPC* (grouped by $|V_{\text{NPC}}| \in \{5k, 5k+1, \ldots, 5k+4\}$) and *instances per given number of edges of the NPC* (grouped by $|E_{\text{NPC}}| \in \{5k, 5k+1, \ldots, 5k+4\}$) for all non-planar Rome (left) and North (right) graphs. 1.D



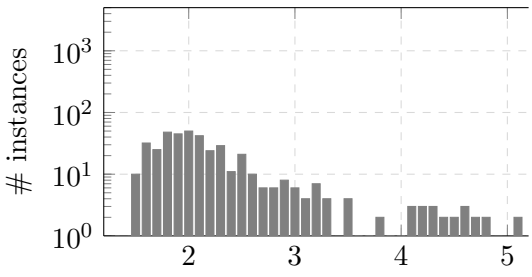
(a) Rome: instances per density



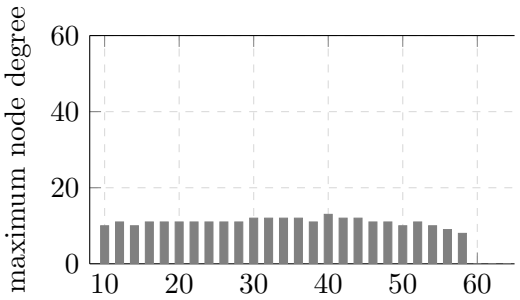
(b) North: instances per density



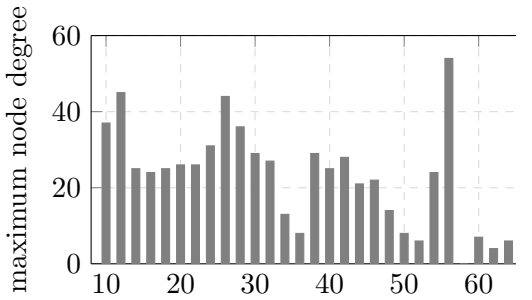
(c) Rome: instances per NPC density



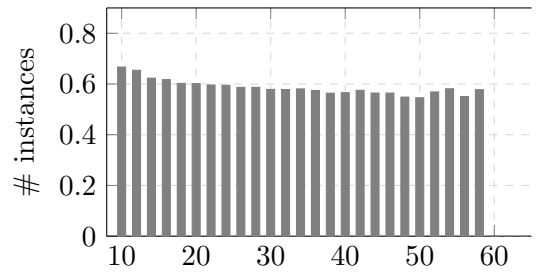
(d) North: instances per NPC density



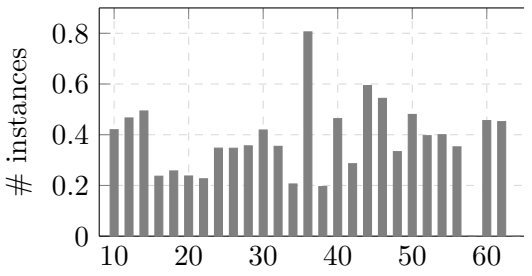
(e) Rome: NPC max. degr. per nodes



(f) North: NPC max. degr. per nodes



(g) Rome: % $d_v = 3$ nodes in NPC



(h) North: % $d_v = 3$ nodes in NPC

1.E **Figure 1.E:** Characteristica *instances per given density* $d_G := |E|/|V|$ (grouped by $d_G \in [0.1{\cdot}k, 0.1{\cdot}(k{+}1)[$), *instances per given NPC density* $d_N := |E_{\text{NPC}}|/|V_{\text{NPC}}|$ (grouped by $d_N \in [0.1 \cdot k, 0.1 \cdot (k+1)[$), *maximum node degree per given number of nodes of the NPC* (grouped by $|V_{\text{NPC}}| \in \{2k, 2k+1\}$) and *percentage of degree-three nodes per given number of nodes of the NPC* (grouped by $|V_{\text{NPC}}| \in \{2k, 2k+1\}$) for all non-planar Rome (left) and North (right) graphs.

Chapter 2

Exact Algorithms for the Minimum Genus Problem

In this chapter we consider the Minimum Genus Problem (MGP) of a graph, see Section 1.2 for the definition. To the best of our knowledge, there are no known formulations for this problem. We propose the first ILP and SAT formulations for this problem. These formulations allow us to develop the first working implementations of general algorithms for the problem, other than exhaustive search.

We investigate several different ways to speed-up and strengthen the formulations; our experimental evaluation on the well-established North and Rome graph instances shows that our approach performs well on small to medium-sized graphs with small genus, and compares favorably to other approaches.

Parts of this chapter were published in [Bey+16] and presented by Michal Kotrbčík at the 15th International Symposium on Experimental Algorithms (2016) in St. Petersburg.

The chapter is structured as follows: First, the basic ideas behind all formulations are presented. Then, an exponentially sized formulation is given followed by polynomially sized variants. After that we focus on five speed-up techniques that are tested and evaluated within a computation framework. Finally, we consider the MGP on non-orientable surfaces and remaining open tasks and problems.

This chapter focuses on the various SAT variants, which correspond to the division of work in the joint project with my co-authors. Most of the ideas can also be applied in our ILP formulations. To underline this, some examples of the according ILP formulations are given.

In addition to the original publication, this chapter includes the *face skipping* and *incremental formulation* speed-ups, the comprehensive comparison of all SAT variants (where we used a greedy approach to find good parameters for the initial publication) and the extension to the non-orientable case.

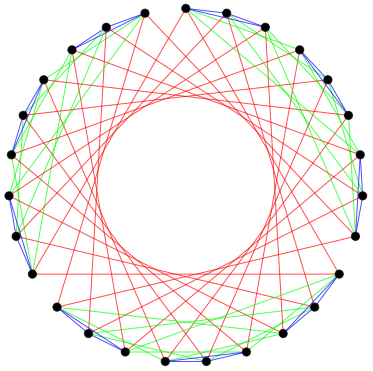
2.1 Introduction

As one of the most important measures of non-planarity, the minimum genus of a graph is of significant interest in computer science and mathematics. However, the problem is notoriously difficult from the theoretical, practical, and also structural perspective. Indeed, its complexity was listed as one of the 12 most important open problems in the first edition of Garey and Johnson's book [GJ79]; Thomassen established its NP-completeness in general [Tho89] and for cubic graphs [Tho97]. Although the existence of an $\mathcal{O}(1)$-approximation can currently not be ruled out, there was no general positive result beyond a trivial $\mathcal{O}(|V|/g)$-approximation until a recent breakthrough by Chekuri and Sidiropoulos [CS13]. For graphs with bounded degree, they

provide an algorithm that either correctly decides that the genus of the graph G is greater than g, or embeds G in a surface of genus at most $g^{\mathcal{O}(1)} \cdot (\log |V|)^{\mathcal{O}(1)}$. Very recently, Kawarabayashi and Sidiropoulos [KS15] showed that the bounded degree assumption can be omitted for the related problem of *Euler genus* by providing an $\mathcal{O}(g^{256}(\log |V|)^{189})$-approximation; however, this does not yield an approximation for orientable genus.

Minimum genus is a useful parameter in algorithm design, since, similarly to the planar case, we can take advantage of the topological structure to design faster algorithms for graphs of bounded genus. However, these algorithms typically assume that the input graph is actually embedded in some surface, as for instance in [EFN12; CCE13]. Therefore, without a practical algorithm providing an embedding in a low-genus surface, these algorithms cannot be effectively implemented.

In the mathematical community, the genus of specific graph families is of interest ever since Ringel's celebrated determination of the genus of complete graphs [Rin74]. Such research often combines numerous different approaches, including computer-aided methods, see, e.g., [CG15; KP15]. However, in practice it often turns out that even determining the genus of a single relatively small graph can be rather difficult [Moh+85; BS88; MPW05; CG15; KP15]. One of the reasons is the large problem space—an r-regular graph with n vertices can have $[(r-1)!]^n$ embeddings. A concrete example is the graph $K_3 \square K_3 \square K_3$ (see Figure 2.A), which has genus 7 and was investigated in [Moh+85; BS88]. It is 6-regular, has 27 vertices, and thus has roughly 10^{56} different embeddings. It is known that complete graphs have exponentially many embeddings of minimum genus; however, the known constructions are nearly symmetric and the problem becomes much more difficult when the minimum genus does not equal the trivial bound from Euler's formula, see, e.g., [KP15] for more details. While it is conjectured that the genus distribution of a graph—the number of its embeddings into each orientable surface—is unimodal (first nondecreasing and then from some point on nonincreasing), very little is known about the structure of the problem space both in theory and practice.



2.A **Figure 2.A:** The graph $K_3 \square K_3 \square K_3$ (= Cayley graph of $\mathbb{Z}_3^3$).

From a slightly different perspective, it has been known for a long time that deciding embeddability in a *fixed* surface is polynomial both for the toroidal [Fil78] and the general case [FMR79; DR91]. In fact, the minimum genus is fixed-parameter tractable as a result of the Robertson-Seymour theorem, since for every surface there are only finitely many forbidden graph minors, and testing for a fixed minor needs only polynomial time [KKR12]. While there is a direct linear-time algorithm deciding embeddability in a fixed surface [Moh96; KMR08], taking any of these algorithms to practice is very challenging for several reasons. First, the naive approach of explicitly testing for each forbidden minor is not viable, since the complete list of forbidden minors is known only for the plane and the projective plane, and the number of minors grows rapidly: for the torus there are already more than 16 629 forbidden minors [Cha02]. Second, Myrvold and Kocay [MK11] reviewed existing algorithms to evaluate their suitability for implementation in order to compute the complete list of forbidden toroidal minors. Unfortunately, they report that [Fil78] contains a "*fatal flaw*", that also appears in the algorithm in [FMR79], and that the algorithm in [DR91] is also "*incorrect*". Myrvold and Kocay conclude that "*There appears to be no way to fix these problems without creating algorithms which take exponential time*" [MK11]. Finally, Mohar's algorithm [Moh96], even in the simpler toroidal case [JMM95], is very difficult to implement correctly (see the discussion in [MK11]). Consequently, there is

currently no correct implementation of any algorithm for the general case of the problem beyond exhaustive search.

It is thus desirable to have an effective and correct implementation of a practical algorithm for the minimum genus. Rather surprisingly, to the best of our knowledge, the approach to obtain practical algorithms via ILP (integer linear program) and SAT (satisfiability) solvers has never been attempted for the minimum genus so far.

2.2 Basic Ideas for SAT and ILP Formulations

Our main idea to solve the MGP is based on the insight by Youngs [You63] that one can iterate over all possible rotation systems of a graph, compute the resulting number of faces and thus compute the minimum genus by maximizing the number of faces. However, the number of rotation systems of a given graph G is $\prod_{v\in V}(d_v-1)!$ which is exponential in the input size of G. Therefore, our concept differs from the one by Youngs in an essential point: we construct a minimum genus rotation system using ILPs or SATs instead of iterating over all rotation systems.

We consider finite undirected graphs and assume w.l.o.g. that all graphs are simple, connected, and have minimum degree 3 (see Corollary 2.2).

To realize our approach, we model the so called *face traversal procedure*. See Section 1.1.3 for a description of the procedure and the basic definitions of embeddings, surfaces and faces. We will use facial walks on non-orientable surfaces later in this chapter. For the orientable case we restrict the general definition to $\lambda \equiv 1$.

Up to mirror images of the surfaces, there is a 1-to-1 correspondence between rotation systems of G and (cellular) embeddings of G into orientable surfaces (see [GT87, Theorem 3.2.3] and [Edm60; Hef91]). Euler's formula asserts that each (cellular) embedding of G in an orientable surface satisfies $|V|-|E|+f = 2-2g$, where f is the number of the faces of the embedding, and g is the genus of the underlying surface. It follows that (i) determining the genus of the underlying surface for a given rotation system is essentially equivalent to calculating the number of faces; and (ii) finding the genus of a graph corresponds to maximizing the number of faces over all rotation systems of the graph. See [MT01] for more details.

Our ILP approaches will maximize the number of faces over the possible rotation systems. In order to construct the ILP we use an upper bound $\bar{f}$ for the number of faces of a given graph. On the other hand, we will use our SAT formulations to solve the question "is there an embedding with at least f faces?"

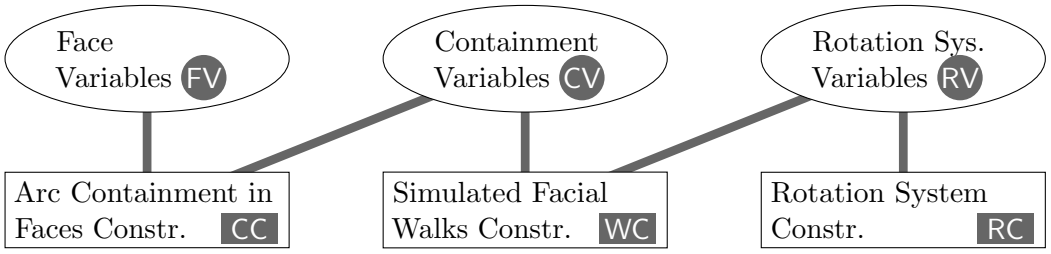
Lemma 2.1. *Let $G=(V,E)$ be a connected, non-planar graph and $f^+ := \min\{\lfloor 2|E|/3\rfloor, |E|-|V|\}$. The number $\bar{f} = f^+ - \left[f^+ \cdot (|E|-|V|) \bmod 2\right]$ is an upper bound for the number of faces of G in an orientable embedding.* 2.1

Proof. Using the Euler characteristic $2-2g=\chi=|V|-|E|+|F|$, we conclude that $|F| = 2-2g+|E|-|V| \le |E|-|V|$, so $|E|-|V|$ is a valid upper bound. Furthermore, imagine that there are more than $2|E|/3$ faces. Obviously, we then have used more than $3(2|E|/3)=|A|$ darts in the facial walks, a contradiction due to $\lambda\equiv 1$.

Using χ again, it follows that $|V|-|E|+|F| \equiv 0 \bmod 2$, so $|F| \equiv |V|-|E| \bmod 2$. Thus, if $|V|-|E| \equiv 0 \bmod 2$ and $f^+ \equiv 1 \bmod 2$ the upper bound f^+ can be improved by 1, and vice versa. □

The formulations we present in the next sections always answer the question: "*is there an embedding of the given graph with at least f faces*?" where f is a parameter. In our formulations we use variables x_i to model whether a face with the index i is used. We use them as follows:

- In our SAT approach we use only $x_1,\dots,x_f$ in the model and set all of them to `true`.



2.B **Figure 2.B:** Interplay of the six blocks of all formulations. A formulation is defined by three variable blocks FV, CV, RV and three constraint blocks CC, WC, RC.

- In our ILP approach we use Use $x_1, \ldots, x_{\bar{f}}$ in the model and

$$\max \sum_{i=1}^{\bar{f}} x_i$$

as objective function.

2.3 Exponentially Sized Formulations: Basic ILP and SAT Models

In this section, we describe how to reformulate the MGP as an integer linear program (ILP) or a related problem of Boolean satisfiability (SAT). We first describe the basic concepts of both formulations, and later consider possible ways to improve them.

All of our formulations consist of (a subset of) the following six building blocks, where each block can be designed/parameterized in several ways (which will be discussed later):

FV face variables, *e.g.*, x_i determines if the face with index i is used;

CV containment variables, *e.g.*, c_a^i determines if dart a is contained in the face with index i;

RV rotation variables, encode the rotation at a vertex;

CC containment constraints, various constraints that *e.g.*, ensure that darts are contained in only one face, Kirchhoff constraints, and further speed-up methods;

WC facial walks constraints, simulate the face traversal procedure;

RC rotation system constraints, ensure that the RV form a feasible rotation system.

The interplay of the six blocks is shown in Figure 2.B. We will write $\mathbb{I}C$, $\mathbb{I}W$, $\mathbb{I}R$ and $\mathbb{B}C$, $\mathbb{B}W$, $\mathbb{B}R$ for CC, WC, RC when want to distinguish between the ILP and SAT formulation, respectively.

First ILP Model. Let $\bar{f}$ be an upper bound on the attainable number of faces. For each $i \in [\bar{f}]$, we have a binary variable x_i that is 1 if and only if the i-th face exists and a binary variable c_a^i, for each $a \in A$, that is 1 if and only if dart a is traversed by the i-th face. For each vertex $v \in V$ and neighbors $u, w \in N(v), u \neq w$, the binary variable $p_{u,w}^v$ is 1 if and only if w is the direct

successor of u in the rotation at v. Our first ILP formulation $\mathbb{I}^{\mathrm{I}}$ then is:

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{\bar{f}} x_i & & & \\
\text{s.t.} \quad & 3x_i \leq \sum_{a \in A} c_a^i & & \forall i \in [\bar{f}] & (\mathbb{I}C_1^{\mathrm{I}}) \\
& \sum_{i=1}^{\bar{f}} c_a^i = 1 & & \forall a \in A & (\mathbb{I}C_2^{\mathrm{I}}) \\
& \sum_{a \in \delta^-(v)} c_a^i = \sum_{a \in \delta^+(v)} c_a^i & & \forall i \in [\bar{f}],\, v \in V & (\mathbb{I}C_3^{\mathrm{I}}) \\
& c_{vw}^i \geq c_{uv}^i + p_{u,w}^v - 1 & & \forall i \in [\bar{f}],\, v \in V,\, u \neq w \in N(v) & (\mathbb{I}W_1^{\mathrm{I}}) \\
& c_{uv}^i \geq c_{vw}^i + p_{u,w}^v - 1 & & \forall i \in [\bar{f}],\, v \in V,\, u \neq w \in N(v) & (\mathbb{I}W_2^{\mathrm{I}}) \\
& \sum_{w \in N(v), u \neq w} p_{u,w}^v = 1 & & \forall v \in V,\, u \in N(v) & (\mathbb{I}R_1^{\mathrm{I}}) \\
& \sum_{u \in N(v), w \neq u} p_{u,w}^v = 1 & & \forall v \in V,\, w \in N(v) & (\mathbb{I}R_2^{\mathrm{I}}) \\
& \sum_{u \in U} \sum_{w \in N(v) \setminus U} p_{u,w}^v \geq 1 & & \forall v \in V,\, \emptyset \neq U \subsetneq N(v) & (\mathbb{I}R_3^{\mathrm{I}}) \\
& x_i, c_a^i, p_{u,w}^v \in \{0,1\} & & \forall i \in [\bar{f}],\, a \in A,\, v \in V,\, u \neq w \in N(v) &
\end{aligned}
$$

Theorem. *Formulation $\mathbb{I}^{\mathrm{I}}$ solves MGP.*

Proof. Constraints $(\mathbb{I}C_1^{\mathrm{I}})$ ensure that if a face exists, it traverses *at least* three darts[1]; inversely, each dart is traversed by exactly one face due to $(\mathbb{I}C_2^{\mathrm{I}})$. Equalities $(\mathbb{I}C_3^{\mathrm{I}})$ guarantee that at every vertex of a face i, the number of i-traversed incoming and outgoing darts is identical. Inequalities $(\mathbb{I}W_1^{\mathrm{I}})$ and $(\mathbb{I}W_2^{\mathrm{I}})$ ensure that darts uv and vw are both in the same face if w is the successor of u in the rotation at v. Constraints $(\mathbb{I}R_1^{\mathrm{I}})$ and $(\mathbb{I}R_2^{\mathrm{I}})$ ensure that p^v represents a permutation of the vertices in $N(v)$; $(\mathbb{I}R_3^{\mathrm{I}})$ ensures that p^v consists of a single cycle. Observe that maximizing the objective function guarantees that each face index corresponds to at most one facial walk.□

Remark. Note that constraints $(\mathbb{I}R_3^{\mathrm{I}})$ are sub-tour elimination constraints typically found in models for the Travelling Salesman Problem. In our context the elimination of sub-tours means that there is no sub-cycle in the rotation around a vertex. Thus, the rotation around each vertex consists of a single cycle of its neighbors.

Furthermore, the formulation has a polynomial number of variables and the *number of constraints is exponential* in $\Delta(G)$. ⌟

First SAT Model. To solve the above ILP, we will need to consider its linear relaxation (where the binary variables are replaced by variables in the interval $[0, 1]$). It is easy to see that fractional values for the p^v matrices lead to very weak dual bounds.[2] Therefore, we also consider SAT formulations. While general SAT solvers cannot take advantage of algebraically obtained (lower) bounds, state-of-the-art SAT solvers are highly tuned to quickly search a vast solution space by sophisticated branching, backtracking, and learning strategies. This can give them an upper hand over ILP approaches, in particular when the ILP's relaxation is weak.

In contrast to the ILP, a SAT problem has no objective function and simply asks for *some* satisfying variable assignment. In our case, we construct a SAT instance to answer the question whether the given graph allows an embedding with *at least* f faces. To solve the optimization problem, we iterate the process for increasing values of f until reaching unsatisfiability.

[1] For a simple graph, the minimum genus embedding contains no face of length 1 or 2. On the other hand, we cannot be more specific than the lower bound of 3.

[2] The values $p_{u,w}^v := 1/(d_v - 1)$ satisfy the *RC* in $\mathbb{I}^{\mathrm{I}}$. Furthermore setting $c_a^i := 1/\bar{f}$ satisfies the *WC* and allows to set all x_i to 1. Hence, the *CC* are satisfied, too. Thus, we have $\sum_{i=1}^{\bar{f}} x_i = \bar{f}$ as value of the objective function.

$\mathbb{I}^{\mathrm{I}}$ & $\mathbb{B}^{\mathrm{I}}$	ILP	one SAT iteration	all SAT iterations
FV	$\bar{f}$	0	0
CV	$\bar{f}\lvert E\rvert$	$f\lvert E\rvert$	$\bar{f}\lvert E\rvert$
RV	$\Delta(G)^2\lvert V\rvert$	$\Delta(G)^2\lvert V\rvert$	$\Delta(G)^2\lvert V\rvert$
CC	$\bar{f}\lvert V\rvert + \lvert E\rvert$	$f^2\lvert E\rvert$	$\bar{f}^3\lvert E\rvert$
WC	$\Delta(G)^2\lvert V\rvert\bar{f}$	$\Delta(G)^2\lvert V\rvert f$	$\Delta(G)^2\lvert V\rvert\bar{f}^2$
RC	$2^{\Delta(G)}\lvert V\rvert$	$2^{\Delta(G)}\lvert V\rvert$	$2^{\Delta(G)}\lvert V\rvert\bar{f}$

2.C **Table 2.C:** Number of variables and constraints for the initial formulations $\mathbb{I}^{\mathrm{I}}$ and $\mathbb{B}^{\mathrm{I}}$. We only give bounds in terms of $\mathcal{O}$-notation.

Remark. In Chapter 4 we use PBS (pseudo-Boolean satisfiability) formulations to model SAT problems with objective functions. Our method of using the following SAT formulations—to find the maximum feasible number f of faces—simulates a PBS solver. In the next sections, we present speed-up methods to equip our models with structures that can benefit from our knowledge, e.g. about the parity described in the lemma above. Therefore, we refrain from using PBS formulations here. ⌟

We use the same notation as before, and construct the SAT formulation around the very same ideas. Each binary variable is now a Boolean variable instead. While a SAT formulation is typically given in conjunctive normal form (CNF), we present it here as a conjunction of separate Boolean formulae (rules) for better readability. Their transformation into equisatisfiable CNFs is trivial in our usecases. The initial SAT formulation $\mathbb{B}^{\mathrm{I}}$ is:

$$\neg(c_a^i \wedge c_a^j) \qquad \forall a \in A,\, i \neq j \in [f] \qquad (\mathbb{B}C_1^{\mathrm{I}})$$

$$\bigvee\nolimits_{a\in A}\, c_a^i \qquad \forall i \in [f] \qquad (\mathbb{B}C_2^{\mathrm{I}})$$

$$p_{u,w}^v \rightarrow (c_{uv}^i \leftrightarrow c_{vw}^i) \qquad \forall v \in V,\, u \neq w \in N(v),\, i \in [f] \qquad (\mathbb{B}W_1^{\mathrm{I}})$$

$$\bigvee\nolimits_{u\in N(v),u\neq w}\, p_{u,w}^v \qquad \forall v \in V,\, w \in N(v) \qquad (\mathbb{B}R_1^{\mathrm{I}})$$

$$\neg(p_{u,w}^v \wedge p_{u',w}^v) \qquad \forall v \in V,\, w \in N(v),\, u \neq u' \in N(v) \setminus \{w\} \qquad (\mathbb{B}R_2^{\mathrm{I}})$$

$$\bigvee\nolimits_{w\in N(v),w\neq u}\, p_{u,w}^v \qquad \forall v \in V,\, u \in N(v) \qquad (\mathbb{B}R_3^{\mathrm{I}})$$

$$\neg(p_{u,w}^v \wedge p_{u,w'}^v) \qquad \forall v \in V,\, u \in N(v),\, w \neq w' \in N(v) \setminus \{u\} \qquad (\mathbb{B}R_4^{\mathrm{I}})$$

$$\bigvee\nolimits_{u\in U,w\in N(v)\setminus U}\, p_{u,w}^v \qquad \forall v \in V,\, \emptyset \neq U \subsetneq N(v) \qquad (\mathbb{B}R_5^{\mathrm{I}})$$

Theorem. *The SAT formulation above solves MGP with respect to f.*

Proof. Rules $(\mathbb{B}C_1^{\mathrm{I}})$ and $(\mathbb{B}C_2^{\mathrm{I}})$ enforce that each dart is traversed by exactly one face and each face contains at least one dart, cf. $(\mathbb{I}C_2^{\mathrm{I}})$. Rule $(\mathbb{B}W_1^{\mathrm{I}})$ ensures that the successor is in the same face, cf. $(\mathbb{I}W_1^{\mathrm{I}})$–$(\mathbb{I}W_2^{\mathrm{I}})$. Rules $(\mathbb{B}R_1^{\mathrm{I}})$–$(\mathbb{B}R_5^{\mathrm{I}})$ guarantee that p^v variables form rotations at v, cf. $(\mathbb{I}R_1^{\mathrm{I}})$–$(\mathbb{I}R_3^{\mathrm{I}})$. □

Characteristica of the initial formulations are shown in Table 2.C. Both formulations have the same number of variables (where we assume that in the worst-case $f = \bar{f}$ for the final SAT iteration). Clearly, solving many SAT iterations consecutively results in a larger number of constraints compared to the ILP formulation.

The performance of our initial formulations is not good enough to build a practical minimum genus computation tool on them. We use them as a basis of comparison for more sophisticated formulations both for run-time benchmarks as well as correctness testing.

In the following sections we discuss reformulations to gain speed-ups on the theoretical side and in practice. We will see polynomially sized formulations that are slower in practice, tighter constraints that do not result in faster algorithms, and vast speed-ups by changing the representation of objects or by simplifications based on special graph structures.

2.4 Polynomially Sized Formulations: Index and Betweenness Reformulation

Remember that the number of inequalities ($\mathbb{I}R_3^{\mathrm{I}}$), or rules ($\mathbb{B}R_5^{\mathrm{I}}$) respectively, is exponential in the degree of each vertex v. Therefore, we investigate ways to obtain a polynomial time solution strategy or a polynomially sized formulation.

Polynomial Time Solution Strategy. For the ILP, we can *separate* violating constraints (also known as row generation) using a well-known separation oracle based on minimum cuts (see, e.g., [Coo+98, Sec. 7.4]). While this guarantees that only a polynomially sized subset of ($\mathbb{I}R_3^{\mathrm{I}}$) is used, it is not worthwhile in practice: First, the maximum degree is typically small; second, the separation process requires a comparably large overhead and third, state-of-the-art ILP solvers offer a lot of speed-up techniques that need to be deactivated to separate constraints on the fly. Overall, this more than doubles the running times compared to a direct inclusion of all ($\mathbb{I}R_3^{\mathrm{I}}$), even if we separate only for vertices with large degrees.

Another option is to use different representations for rotation systems. Here we discuss an *index* approach and a *betweenness* approach. Both yield polynomially sized formulations.

Index Reformulation. For the index approach we replace the permutation variables with variables that attach vertices to specific positions in the rotation. This is known to be weaker in the realm of ILPs, and we hence concentrate on the SAT formulation. There, we introduce for any $v \in V, u \in N(v)$ a Boolean variable $q_{j,u}^v$ that is `true` if and only if u is the j-th vertex in the rotation at v. We do not use the p variables any longer, replace the old permutation rules ($\mathbb{B}R_1^{\mathrm{I}}$)–($\mathbb{B}R_5^{\mathrm{I}}$) with rules to ensure that each q^v is a bijective mapping:

$$\bigvee\nolimits_{j\in[d_v]} q_{j,u}^v \qquad \forall v \in V,\, u \in N(v) \qquad (\mathbb{B}R_1^{\mathrm{Ind}})$$

$$\neg(q_{j,u}^v \wedge q_{k,u}^v) \qquad \forall v \in V,\, u \in N(v),\, j \neq k \in [d_v] \qquad (\mathbb{B}R_2^{\mathrm{Ind}})$$

$$\neg(q_{j,u}^v \wedge q_{j,w}^v) \qquad \forall v \in V,\, j \in [d_v],\, u \neq w \in N(v) \qquad (\mathbb{B}R_3^{\mathrm{Ind}})$$

and change ($\mathbb{B}W_1^{\mathrm{I}}$) to

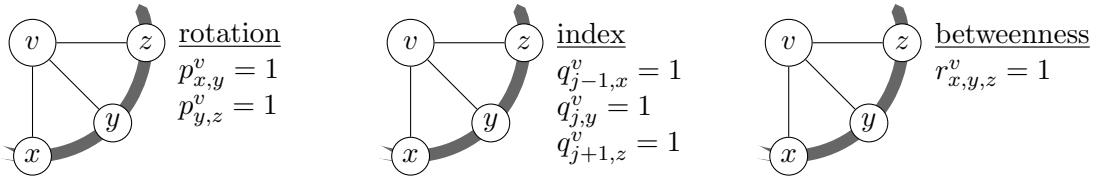
$$\bigvee\nolimits_{j\in[d_v]} (q_{j,u}^v \wedge q_{j+1,w}^v) \rightarrow (c_{uv}^i \leftrightarrow c_{vw}^i) \quad \forall v \in V,\, u \neq w \in N(v),\, i \in [f]. \qquad (\mathbb{B}W_1^{\mathrm{Ind}})$$

The effect of the reformulation will be studied in detail in Section 2.7. In short, the number of solved instances falls and the run-time simultaneously increases.

Betweenness Reformulation. For the betweenness approach we add the variables $r_{x,y,z}^v$ for each triple of pairwise distinct $x, y, z \in N(v)$. By $r_{x,y,z}^v = 1$ (`true`, respectively) we denote that y is (somewhere) between x and z in the counter-clockwise rotation at v. Here we only describe the

	$\mathbb{B}^{\mathrm{I}}$	$\mathbb{B}^{\mathrm{Ind}}$	$\mathbb{B}^{\mathrm{Bet}}$
RV	$\Delta(G)^2\|V\|$	$\Delta(G)^2\|V\| + \|E\|$	$\Delta(G)^3\|V\|$
WC	$\Delta(G)^2\|V\|f$	$\Delta(G)^2\|V\|f$	$\Delta(G)^2\|V\|f$
RC	$2^{\Delta(G)}\|V\|$	$\Delta(G)^3\|V\|$	$\Delta(G)^4\|V\|$

2.D **Table 2.D:** Number of variables and constraints of the ordering reformulations compared to the initial formulation. We only give bounds in terms of $\mathcal{O}$-notation.



2.E **Figure 2.E:** Example variable assignment for the three different representation of the rotation around vertex v. The shown neighbors appear as $\cdots \to x \to y \to z \to \cdots$ in the cyclic order. Note that we would have $r^v_{x,y,z} = 1$ even if the neighbors appear as $\cdots \to x \to u \to y \to z \to \cdots$ in the cyclic order.

usage of the r variables in the SAT formulation. The usage in the ILP is analogous. First of all, the cyclicity of a rotation implies the symmetries

$$r^v_{x,y,z} \equiv r^v_{y,z,x} \equiv r^v_{z,x,y} \equiv \neg r^v_{x,z,y} \equiv \neg r^v_{z,y,x} \equiv \neg r^v_{y,x,z} \quad \forall\{x,y,z\} \subseteq N(v). \tag{$\mathbb{B}R_1^{\mathrm{Bet}}$}$$

Note that from an implementation point of view, it is possible to merge the different variables $r^v_{x,y,z}$, $r^v_{y,z,x}$, $r^v_{z,x,y}$, $r^v_{x,z,y}$, $r^v_{z,y,x}$, $r^v_{y,x,z}$ into a single variable $r^v_{\{x,y,z\}}$ by using Equations ($\mathbb{B}R_1^{\mathrm{Bet}}$) directly in the other constraints.

Instead of ensuring that each p^v represents a permutation, we connect the p-variables to the new r-variables via

$$p^v_{u,w} \leftrightarrow \bigwedge\nolimits_{y \in N(v)\setminus\{u,w\}} r^v_{u,w,y} \qquad \forall v \in V,\, u \neq w \in N(v) \tag{$\mathbb{B}R_2^{\mathrm{Bet}}$}$$

Note that we keep the constraints ($\mathbb{B}R_1^{\mathrm{I}}$)–($\mathbb{B}R_5^{\mathrm{I}}$) and replace only RC. The rules to model the betweenness conditions for the neighborhood of a given vertex v are simply

$$r^v_{u,w,x} \wedge r^v_{u,x,y} \to r^v_{u,w,y} \wedge r^v_{w,x,y} \qquad \forall\{u,w,x,y\} \subseteq N(v) \tag{$\mathbb{B}R_3^{\mathrm{Bet}}$}$$

The effect of the reformulation will be studied in detail in Section 2.7. Essentially, the results are similar to the index reformulation above.

Summary and Conclusion. The three different approaches to model the rotation around each vertex are summarized in Figure 2.E. Table 2.D shows the number of required variables and constraints.

By using the rotation variables p^v we have a direct representation of successor and predecessor. This way it is easy to cast the facial walk algorithms into constraints. However, we need $\mathcal{O}(2^{|V|})$ cut constraints to ensure that the rotation around each vertex consists of exactly one cycle. On a small test instance with 405 random samples from the Rome graph library the initial SAT formulation solved 108 instances in an average time of 27 seconds on the successful instances.

The index based approach is not as easy to handle. Each time we have to check if there is an index $j \in [d_v]$ such that the pair $(j, j+1)$ represents the current successor or predecessor that

we are looking for. The number of RC constraints drops to $\mathcal{O}(|V|^4)$. However, on our small test set we solved only 102 instances; the common set solved both by $\mathbb{B}^{\mathrm{I}}$ and $\mathbb{B}^{\mathrm{Ind}}$ has 100 instances. The run-time on the common set increased to 97.5 seconds in the average case.

Finally, we look at the betweenness formulation. The number of RC constraints has dropped to $\mathcal{O}(|V|^5)$ compared to the initial formulation. As in the index-based reformulation, we need auxiliary constraints ($\mathbb{B}R_2^{\mathrm{Bet}}$) to connect the betweenness relation to the successor/predecessor relation that we use in the other constraints. On the small test set we solved only 104 instances; the common set solved both by $\mathbb{B}^{\mathrm{I}}$ and $\mathbb{B}^{\mathrm{Bet}}$ has slightly different but again 100 instances. The run-time on the common set increased to 97.6 seconds in the average case.

A direct comparison between $\mathbb{B}^{\mathrm{Ind}}$ and $\mathbb{B}^{\mathrm{Bet}}$ shows that the commonly solved set has a size of 99 instances and the average run-time on this set does not differ significantly.

Overall, we conclude that the exponential dependencies of the original formulations are not so much of an issue in practice after all, and the overhead and weaknesses of polynomial strategies typically do not seem worthwhile. However, if one considers problems with many very high degree vertices where the exponential dependency becomes an issue, the above approaches can be merged very naturally, leading to an overall polynomially sized model: Let ϑ be some fixed constant threshold value (to be decided upon experimentally). For vertices v of degree at most ϑ, we use the original formulation requiring an exponential (in constant ϑ) number of constraints over p^v. Vertices of degree above ϑ are handled via the betweenness reformulation.

In our experiments (see Section 2.7) we see that only ten instances (2 %) of the North graph library have a degree distribution that prohibits the use of our exponentially sized formulations.

2.5 Speed-Up Techniques

The methods presented in the previous section yield polynomially sized formulations. But they result in an increased run-time and a decreased success rate. There are several potential opportunities to improve upon the initial formulation. In pilot studies we investigated their practical ramifications.

Symmetries. In our initial formulation $\mathbb{I}^{\mathrm{I}}$ we have some symmetries that needlessly increase the search space for feasible solutions, and thus also the branch-and-cut search tree. Here we discuss two examples.

1. Objective function: When the upper bound is $\bar{f} = 10$ and the optimal solution has, say, $f = 8$ faces, there are $\binom{10}{8} = 45$ possibilities to fill 8 spots in the objective function $x_0 + \cdots + x_9$ with 1's. The symmetry-breaking constraints

$$x_i \geq x_{i-1}, \qquad i = 0, \ldots, \bar{f} - 2$$

   prohibit this behavior.

2. Face sizes: Let $|f_i|$ denote the size (number of darts) of face f_i. We can restrict the search space to feasible instances with $|f_0| \geq |f_1| \geq \ldots \geq |f_{\bar{f}-1}|$. This can be done by

$$\sum\nolimits_{a \in A} c_a^i \geq \sum\nolimits_{a \in A} c_a^{i+1}, \qquad i = 0, \ldots, \bar{f} - 2.$$

Surprisingly, this does not improve the overall ILP running time (and the latter is even worse by orders of magnitude), and we refrain from using these constraints in the following.

	$\mathbb{I}^{\mathrm{I}}$ & $\mathbb{I}^{\mathrm{D3}}$	$\mathbb{B}^{\mathrm{I}}$ & $\mathbb{B}^{\mathrm{D3}}$
RV	$\lvert V_3\rvert + \Delta(G)^2 \lvert V \setminus V_3\rvert$	$\lvert V_3\rvert + \Delta(G)^2 \lvert V \setminus V_3\rvert$
WC	$(\lvert V_3\rvert + \Delta(G)^2 \lvert V \setminus V_3\rvert)\bar{f}$	$(\lvert V_3\rvert + \Delta(G)^2 \lvert V \setminus V_3\rvert)f$

2.F **Table 2.F:** Number of variables and constraints of the low-degree vertices reformulations compared to the initial formulation. We only give bounds in terms of $\mathcal{O}$-notation.

Low-Degree Vertices. Consider a vertex v of degree two and its neighbors $N(v) = \{u_0, u_1\}$. There is only one possibility for the rotation around v: $u_0 \to u_1 \to u_0$. Thus, we do not need variables for the rotation system of such vertices. However, we do not even consider such vertices, since we can restrict ourselves to minimum vertex degree three (cf. Corollary 2.2).

But a similar observation holds for degree-three vertices. Let $V_3 := \{v \in V : d_v = 3\}$. Consider a degree-three vertex $v \in V_3$ with neighbors u_0, u_1, u_2. The only two possible rotations at v are $u_0 \to u_1 \to u_2 \to u_0$ and $u_2 \to u_1 \to u_0 \to u_2$. Hence, we can use a single binary/Boolean variable p^v whose assignment represents this choice.

In the ILP, we remove all $p^v_{u,w}$ variables for $v \in V_3$, delete the according $\mathbb{I}R^{\mathrm{I}}$ constraints, and replace $\mathbb{I}W^{\mathrm{I}}$ (on V_3) by

$$c^i_{vu_{k+1}} \geq c^i_{u_kv} + p^v - 1 \quad \forall i \in [\bar{f}],\, v \in V_3,\, k \in [3] \qquad (\mathbb{I}W_1^{\mathrm{D3}})$$
$$c^i_{u_kv} \geq c^i_{vu_{k+1}} + p^v - 1 \quad \forall i \in [\bar{f}],\, v \in V_3,\, k \in [3] \qquad (\mathbb{I}W_2^{\mathrm{D3}})$$
$$c^i_{vu_k} \geq c^i_{u_{k+1}v} - p^v \quad \forall i \in [\bar{f}],\, v \in V_3,\, k \in [3] \qquad (\mathbb{I}W_3^{\mathrm{D3}})$$
$$c^i_{u_{k+1}v} \geq c^i_{vu_k} - p^v \quad \forall i \in [\bar{f}],\, v \in V_3,\, k \in [3], \qquad (\mathbb{I}W_4^{\mathrm{D3}})$$

where u_0, u_1, u_2 denote the arbitrarily but statically ordered neighbors of $v \in V_3$. Remember that in [3] we have $2 + 1 = 0$, so the "$k \in [3]$" covers all cases.

In the SAT formulation, we analogously replace $(\mathbb{B}W_1^{\mathrm{I}})$ on V_3 by

$$p^v \to (c^i_{u_kv} \leftrightarrow c^i_{vu_{k+1}}) \quad \forall v \in V_3,\, k \in [3],\, i \in [f] \qquad (\mathbb{B}W_1^{\mathrm{D3}})$$
$$\neg p^v \to (c^i_{u_{k+1}v} \leftrightarrow c^i_{vu_k}) \quad \forall v \in V_3,\, k \in [3],\, i \in [f]. \qquad (\mathbb{B}W_2^{\mathrm{D3}})$$

The number of constraints is nearly the same as in the initial formulation, but the number of variables is decreased by $5|V_3|$. Details can be found in Table 2.F.

As expected, this is faster by orders of magnitude for certain families of graphs, especially for instances with many degree-three vertices. On the real world *Rome* benchmark set (see Section 2.7), the performance improves by about 10 % for both the ILP and the SAT formulations, compared to their respective formulations with purely $p^v_{u,w}$ variables. As we will see in Section 2.7, the impact of this speed-up method depends on the choice of the other speed-up methods. There are combinations where using the special V_3-variables improves the performance but also cases where the performance drops.

This idea can be generalized for vertices v of arbitrary fixed degree $d_v \geq 4$. There are $\varrho := (d_v - 1)!$ different rotations. Instead of using $\mathcal{O}(d_v^2)$ many variables $p^v_{u,w}$, we introduce $\lceil \log_2 \varrho \rceil$ binary variables and represent the index of the rotation as a binary number. Since this process is coupled with a substantial trade-off of more complicated and weaker constraints, we refrain from using it for $d_v \geq 4$.

Binary Face Representations. A big disadvantage of the initial formulation is that we need a large number of constraints to ensure that each dart is only contained in one face. In the ILP

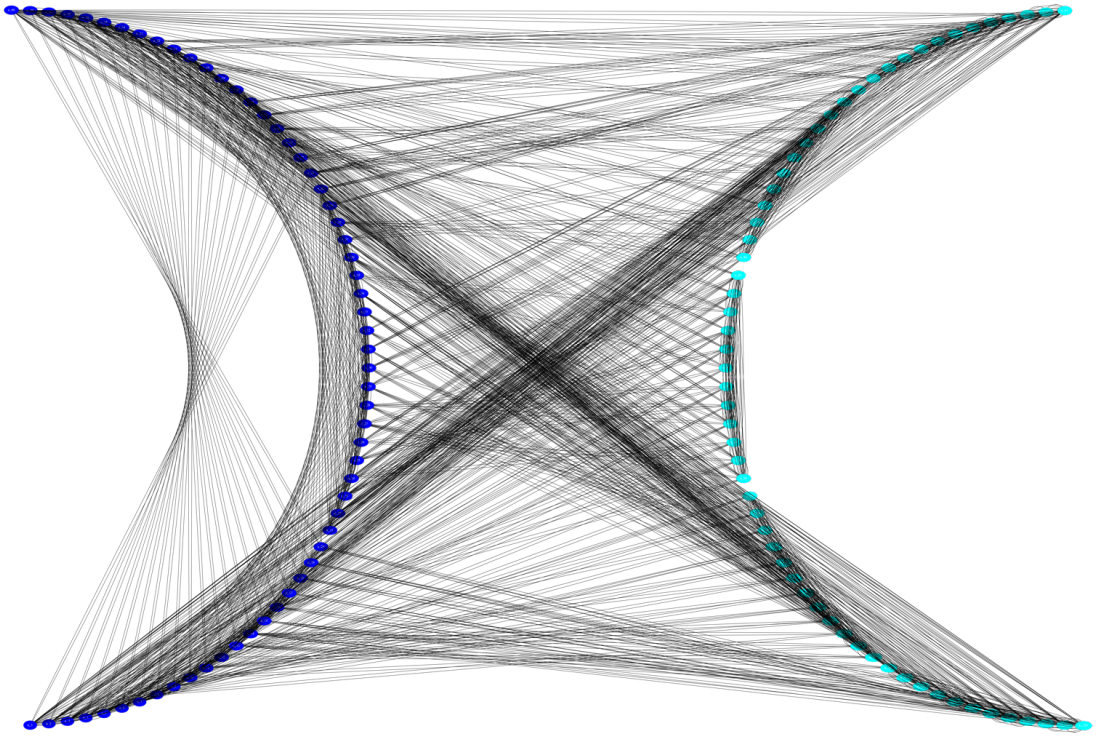


Figure 2.G: Graph of the variables and constraints of $\mathbb{B}^{\mathrm{I}}$ for the input graph K_5. The vertices on the left side are the CV variables c_a^i. The vertices on the right side are the RV variables $p_{u,w}^v$. An edge represents that two variables appear together in one constraint. It is obvious that we have (too) many edges connecting vertices on the left side. The binary face representation attacks this problem. 2.G

variant $\mathbb{I}^{\mathrm{I}}$ this is quite easy:

$$\sum\nolimits_{i=1}^{\bar{f}} c_a^i = 1 \quad \forall a \in A, \tag{$\mathbb{I}C_1^{\mathrm{I}}$}$$

which are only $2|E|$ constraints. But in the SAT formulation $\mathbb{B}^{\mathrm{I}}$ we need

$$\neg(c_a^i \wedge c_a^j) \qquad \forall a \in A,\, i \neq j \in [f], \tag{$\mathbb{B}C_1^{\mathrm{I}}$}$$

which are $f(f-1)|E|$ constraints.

Example. Consider the complete graph K_5 on five vertices. In $\mathbb{I}^{\mathrm{I}}$ we need 20 constraints of this type. The graph has genus one, thus we use $f = \bar{f}$ in our last iteration. This results in 200 constraints in $\mathbb{B}^{\mathrm{I}}$. The example is illustrated in Figure 2.G. ⌟

We attack this problem in the SAT formulation by using an alternative representation of the face indices. Let $i \in [f]$ be a face index, and $\mathcal{B}(i)$ the vector of its binary representation, i.e., $i = \sum_{j=0}^{\ell} 2^j \cdot \mathcal{B}(i)_j$, where $\ell = \lceil \log_2 f \rceil$. We define new Boolean variables b_a^j that are `true` if and only if dart a is contained in a face i with $\mathcal{B}(i)_j = 1$. In logic formulae, value $\mathcal{B}(i)_j = 1$ is naturally mapped to `true`, 0 to `false`.

By changing the clauses ($\mathbb{B}C_2^{\mathrm{I}}$) and ($\mathbb{B}W_1^{\mathrm{I}}$) of the initial SAT formulation above, we construct a new formulation that asks for a solution with at least f faces, because we do not forbid the usage of binary representations outside of $[f]$.

$$\bigvee\nolimits_{a \in A} \bigwedge\nolimits_{j \in [\ell]} (b_a^j \leftrightarrow \mathcal{B}(i)_j) \quad \forall i \in [f] \tag{$\mathbb{B}C_2^{\mathrm{Bin}}$}$$

$$p_{u,w}^v \rightarrow (b_{uv}^j \leftrightarrow b_{vw}^j) \qquad \forall v \in V,\, u \neq w \in N(v),\, j \in [\ell] \tag{$\mathbb{B}W_1^{\mathrm{Bin}}$}$$

	$\mathbb{B}^{\mathrm{I}}$	$\mathbb{B}^{\mathrm{Bin}}$
CV	$\lvert E\rvert f$	$\lvert E\rvert \log f$
CC	$f^2\lvert E\rvert$	f
WC	$\Delta(G)^2\lvert V\rvert f$	$\Delta(G)^2\lvert V\rvert \log f$

2.H **Table 2.H:** Number of variables and constraints of the binary face representation reformulation compared to the initial formulation. We only give bounds in terms of $\mathcal{O}$-notation.

Note that we previously also constructed SAT instances that ask for a solution with at least f faces, but this time we also allow faces with indices up to $2^{\lceil \log_2 \bar{f} \rceil}$. Assume that $\bar{f} = 9$, then we allow face indices from 1 up to $2^{\lceil \log_2 9 \rceil} = 2^4 = 16$. But rule ($\mathbb{B}C_2^{\mathrm{Bin}}$) forces us to use at least f faces. The final solution is nonetheless correct because we will have a no-instance when asking for too many.

The number of variables and constraints is shown in Table 2.H.

The combination of this idea with the low-degree reformulation is straightforward. For $v \in V_3$, we replace ($\mathbb{B}W_1^{\mathrm{Bin}}$) by

$$p^v \rightarrow (b^j_{u_k v} \leftrightarrow b^j_{v u_{k+1}}) \quad \forall v \in V_3,\, k \in [3],\, j \in [\ell] \qquad (\mathbb{B}W_1^{\mathrm{Bin,D3}})$$

$$\neg p^v \rightarrow (b^j_{u_{k+1} v} \leftrightarrow b^j_{v u_k}) \quad \forall v \in V_3,\, k \in [3],\, j \in [\ell]. \qquad (\mathbb{B}W_2^{\mathrm{Bin,D3}})$$

This variant achieves a more than 100-fold speedup for certain families of graphs. On our test set we solved 150 instead of 108 instances in just 1.8 seconds instead of 33 seconds in the average case (over the common set of 108 instances).

Face Skipping. We have two kinds of face indices we do not need to check for feasibility in the SAT variants:

1. Skipping faces by parity: By Euler's formula, we only have to calculate SAT instances with $f \equiv |E| - |V| \mod 2$. For increasing numbers of faces we compute the satisfiability until we get the first unsatisfiable instance. Such an iteration is clearly not necessary in the ILP approach, where our objective function explicitly maximizes f and we only require an upper bound of $\bar{f}$ adjusted for parity. See Lemma 2.1.

2. Skipping faces that are implicitly constructed by a SAT solution asking for an embedding with fewer faces. We will use the following part of this paragraph to discuss this idea in detail.

Consider an iteration where we ask for an embedding with at least f faces. If this is a yes-instance, the SAT solver only knows that there is an embedding with at least f faces. However, it is possible that the rotation system induced by the solution of this yes-instance has more than f faces. This is the case when the SAT formulation labels disjoint (face) cycles in the input graph with the same index. Algorithm 2.I shows an easy way to use this extraction of the realized faces.

An example is shown in Figure 2.J, where we compute the minimum genus of the K_5 by finding an embedding with 5 faces. Without face skipping we see that 1, 3, and 5 are yes-instances and $f = 7$ would be a no-instance. Using $\bar{f} = 5$ we would not check the $f = 7$ instance. If the result of the SAT solver asking for $f = 1$ face induces a rotation system with, say, 3 faces we also would not check the $f = 3$ instance.

The impact of this method highly depends on the graph class. There are settings when using the face skipping we solve 101 instead of 98 instances, coupled with a run-time increasement by

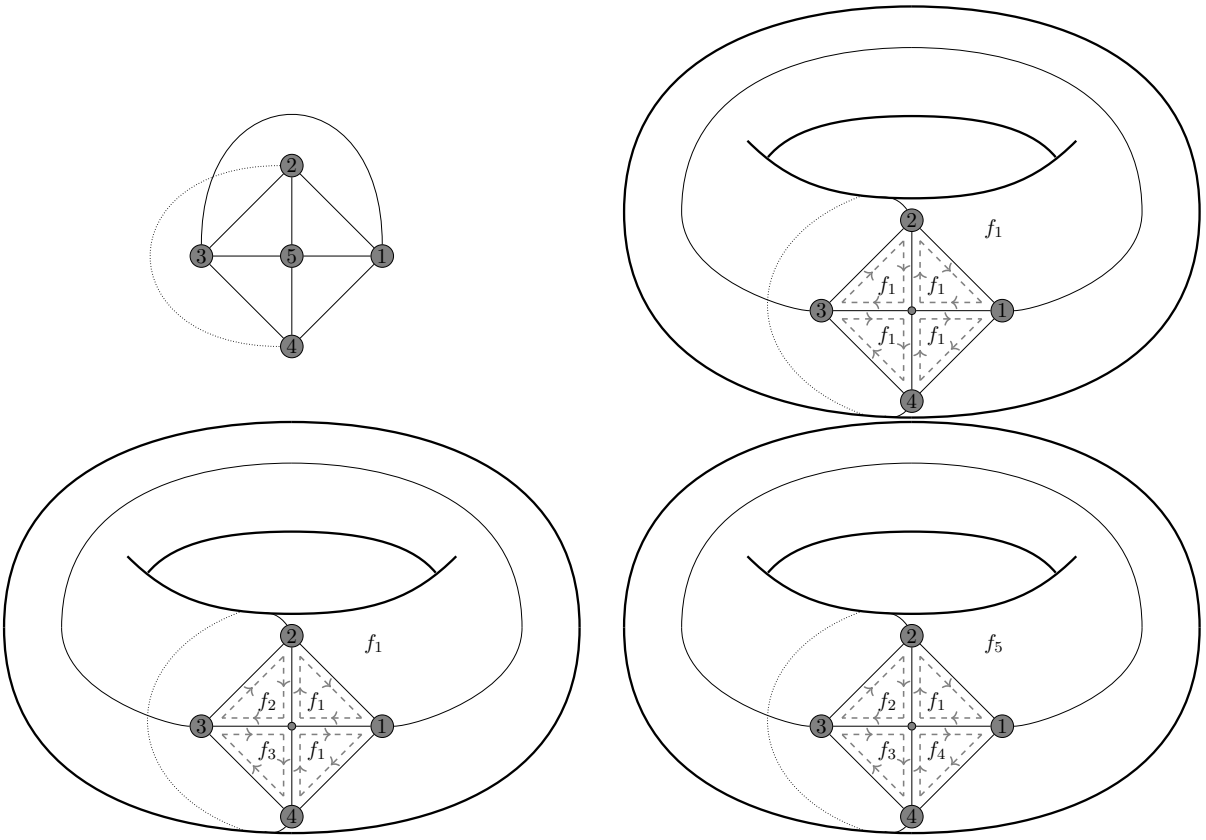


Figure 2.J: The results of iterative SAT solver calls on the input graph K_5. The complete graph on 5 vertices is given on the top left. Now assume that we ask the SAT formulation to check if there is an embedding with at least one face. It is possible that the rotation system that we get from the corresponding variable assignment is a rotation system such that the graph can be embedded on the torus without a crossing. 2.J

Remember our SAT formulation, we only ask for *at least one face*. Thus, our model could result in an optimal rotation system but the SAT formulation labels all 5 faces with the same label "f_1". The reason is that we do not forbid disjoint cycles in a facial walk. The result is shown on the top right: The rotation system is optimal (induces 5 faces) but the SAT solver is not aware of that fact.

In the next iteration we would ask for a solution with at least 3 faces (shown on the bottom left). Again, the rotation system could induce a solution with 5 faces.

In the final iteration (bottom right) we get an optimal solution.

If we use face skipping, the process is a lot faster: We ask for a solution with at least one face. The SAT constructs such a solution. We extract the rotation system from the variable assignment and compute the facial walks by ourselves. This way we immediately see that we already have a solution with 5 faces, which is the optimum in our example.

The faces in the optimal solution are:
f_1: $1 \to 5 \to 2 \to 1$
f_2: $2 \to 5 \to 3 \to 2$
f_3: $3 \to 5 \to 4 \to 3$
f_4: $4 \to 5 \to 1 \to 4$
f_5: $1 \to 2 \to 4 \to 1 \to 3 \to 4 \to 2 \to 3 \to 1$

2.I **Algorithm 2.I:** MinGenus(Graph G)

```
f := 1;
adjust f for parity;
while SAT instance for G to find an embedding with at least f faces is a yes-instance do
    compute the number f_res of realized faces with the facial walk algorithm;
    adjust f_res for parity;
    f := f_res + 2;
return f_res
```

56 percent. In other SAT variants we solve 144 instead of 145 instances but we need only half as long as before. Details follow later.

Incremental Formulations. State-of-the-art SAT solvers are able to perform *warm starts*, which means that we can add further variables and constraints and check for satisfiability of the extended formulation. The solver can use knowledge from the previous calls to solve the extended formulation faster than by starting from scratch.

Our problem is ideally suited for this method: Consider the initial SAT formulation $\mathbb{B}^{\mathrm{I}}$ for a fixed f. To check if the formulation for $f+1$ is also a yes-instance, we need to add the variables c_a^{f+1} for $a \in A$ and the constraints

$$\neg(c_a^i \wedge c_a^{f+1}) \qquad \forall a \in A,\, i \in [f] \qquad (\mathbb{B}C_1^{\mathrm{Incr}:f+1})$$

$$\bigvee\nolimits_{a\in A} c_a^{f+1} \qquad (\mathbb{B}C_2^{\mathrm{Incr}:f+1})$$

$$p_{u,w}^v \rightarrow (c_{uv}^{f+1} \leftrightarrow c_{vw}^{f+1}) \quad \forall v \in V,\, u \neq w \in N(v) \qquad (\mathbb{B}W_1^{\mathrm{Incr}:f+1})$$

Remember that we use the parity argument, so we simultaneously add the variables and constraints above for $f+1$ and $f+2$.

As before, the impact highly depends on the graph class. There are settings when using the incremental formulation we solve 147 instead of 135 instances, coupled with a run-time increase by 9 percent. In other SAT variants we solve the same number of instances but we need only half as long as before. Details follow later.

2.6 A Minimum Genus Computation Framework

Before employing any of our approaches on a given graph, we consider several preprocessing steps. The whole computation can be described as follows:

(S1) Compute the biconnected components B_i of the input graph.

(S2) Compute the unweighted non-planar core C_i of each non-planar component B_i.

(S3) Use one of our approaches to compute the minimum genus $\gamma(C_i)$ of each C_i.

(S4) The minimum genus of the input graph is then $\sum_i \gamma(C_i)$.

Steps (S1) and (S4) is based on an old result by Battle et al. For each biconnected component we compute the genus independently.

Lemma. [Bat$^+$62, Theorem 1], [Arc86][3] *The genus is additive over biconnected components.*

[3] Archdeacon [Arc86] addresses a more general decomposition: A *k-amalgamation* is a graph G that is decomposed into subgraphs G_i such that $G_1 \cup G_2 = G$ and $|V(G_1) \cap V(G_2)| = k$. We write $G_1 \cup_k G_2 = G$. In this context the result of Battle et al. [Bat$^+$62] is $\gamma(G_1 \cup_1 G_2) = \gamma(G_1) + \gamma(G_2)$.

For step (S2) we need to prove that the non-planar core reduction is invariant for our problem. The following theorem is an equivalent of Lemma 1.4 for the minimum genus instead of skewness:

Theorem. *Let G be a 2-connected graph and C be its non-planar core. Then, $\gamma(G) = \gamma(C)$.*

Proof. $\gamma(G) \leq \gamma(C)$: Given an optimal solution for C, we can embed each maximal planar 2-component S onto the surface in place of its replacement edge, without any crossings.

$\gamma(C) \leq \gamma(G)$: Each replaced maximal planar 2-component S contains a path connecting its poles that is drawn crossing-free in the optimal embedding of G; we can planarly draw all of S along this path, and then simplify the embedding by replacing this locally drawn S by its replacement edge; this gives a solution for C on the same surface. □

The result above allows us to restrict the computations to non-planar components. We can test $\gamma(B_i) = 0$ by simply running a linear time planarity test, in our case [BM04]. The unweighted non-planar core itself is also computed in linear time by simply replacing each maximal planar 2-components with an edge.

Corollary 2.2. *Computations for the MGP can be restricted to simple biconnected graphs with minimum degree at least 3.* □ 2.2

2.7 Experimental Evaluation: Different Formulations, Overall Practicality, and Comparison to Existing Genus Computations

In the sections above we developed 48 different SAT variants in total, which arise by combining the variants for each aspect. Our goal is to decide which variant is best to use in practice. This, of course, highly depends on the considered graph class in a specific application.

In the next paragraph we compare all SAT variants against each other on a subset of the Rome graph instances to find the best parameter choice for real world graphs. The selected parameter setting will be used later in this section for all further experiments. We do not present the according results for the ILP formulations as they are not in the focus of our work in this context.[4] The parameter setting selected for the ILP equals the setting for the SAT-based minimum genus algorithm.

In the second part of our experiments we look at the overall practicality of our SAT- and ILP-based algorithms. We examine the success rate (minimum genus computation finished within a fixed amount of time, in our case 20 minutes) and run-time on all non-planar Rome and North graphs. Additionally, we compare both approaches against each other.

Finally, we compare our new approaches to existing minimum genus computations.

SAT Variants. Now, we compare all 48 SAT variants from $R \times D \times C \times F \times I$, where

$$\begin{aligned} R &:= \{\text{rotation system} : \text{initial}, \text{index}, \text{between}\} =: \{\text{ro}_\text{R}, \text{ro}_\text{I}, \text{ro}_\text{B}\} \\ D &:= \{d_v = 3 : \text{yes}, \text{no}\} =: \{\text{cu}_\text{Y}, \text{cu}_\text{N}\} \\ C &:= \{\text{incremental} : \text{yes}, \text{no}\} =: \{\text{in}_\text{Y}, \text{in}_\text{N}\} \\ F &:= \{\text{face skipping} : \text{yes}, \text{no}\} =: \{\text{sk}_\text{Y}, \text{sk}_\text{N}\} \\ I &:= \{\text{face indices} : \mathbb{B}, \mathbb{N}\} =: \{\text{bi}_\text{Y}, \text{bi}_\text{N}\} \end{aligned}$$

against each other. We use a subset of the non-planar Rome instances. We took a random but fixed sample of five instances per node size $\in \{20, 21, \ldots, 100\}$, so we worked on 405 instances to compare all variants. Working on all 8249 non-planar Rome instances would require too much time. We discuss two natural questions:

[4] The development and experimental exploration of the ILP algorithms were done by Stephan Beyer.

1. What is the best parameter setting?

2. Are the choices of the parameters orthogonal to each other?

Our C++ code is compiled with GCC 4.8.5, and runs on a single core of an Intel(R) Xeon(R) CPU E5-2420 v2 with 192 GB DDR3 Memory @ 1600 MHz under Ubuntu 14.04. We use the SAT solver lingeling (improved version for SMT Competition 2015 by Armin Biere)[5], the Open Graph Drawing Framework [Chi+13], and do not apply a virtual memory limit but a time limit of 20 minutes.

In the Tables 2.K and 2.L we use the abbreviations $\mathrm{bi_Y}$ for *binary face representation* and $\mathrm{bi_N}$ for the initial representation; $\mathrm{cu_Y}$ for special treatment of *degree-three* vertices and $\mathrm{cu_N}$ for the initial formulation; $\mathrm{ro_R}$ for the *rotation system* variant using $p^v_{u,w}$ variables, $\mathrm{ro_I}$ using $q^v_{j,u}$, and $\mathrm{ro_B}$ using $r^v_{x,y,z}$; $\mathrm{in_Y}$ for the *incremental formulation* and $\mathrm{in_N}$ for the initial formulation; $\mathrm{sk_Y}$ for the *face skipping* speed-up and $\mathrm{sk_N}$ for the initial method. All tables show the number of solved instances for each variant, the size of the common set (solved by both), the average time on the common set for each variant, and difference in solved instances and run-time.

Table 2.K(a) compares the binary face index representation in all variants. The impact is huge: independent of the choice of the other parameters it is always better to use the binary representation. The number of solved instances increases by approx. 40 % whereas the run-time drops down to approx. 8 % compared to the variants that do not use the binary face representation.

Table 2.K(b) compares the usage of special variables and constraints for degree-three vertices in all variants. The result is not as clear as in the table for the binary face representation. There are parameter choices which double the run-time and lower the number of solved instances, as well as parameter choices with opposite behavior.

Table 2.K(c) compares the incremental formulations that use solver warm starts in all variants. With a few exceptions we almost always increase the number of solved instances and reduce the run-time.

Table 2.K(d) compares the face skipping method in all variants. The results are intermingled, but if we focus on $\mathrm{bi_Y}$ we have a clear trend. We solve almost as many instances as without face skipping, but we safe a good portion of run-time.

Table 2.L(a) compares the effect of using the betweenness formulation instead of the initial formulation (for rotation systems) in all SAT variants. It is clear that we should refrain from using the betweenness variant as it increases the run-time and lowers the number of solved instances.

Table 2.L(b) compares the index reformulation with the initial formulation (for rotation systems) in all SAT variants. We should refrain from using the index reformulation for the same reasons as for the betweenness reformulation. Especially, when we focus on $\mathrm{bi_Y}$ the run-time is more than doubled with almost no effect on the number of solved instances.

Table 2.L(c) compares the betweenness reformulation with the index reformulation. The result is mixed. Restricted to $\mathrm{bi_Y}$ we gain time but also lose some of the solved instances.

The parameter choice we deduce from the experiments is

[5]The previous version was the winner of the *Sequential Appl. SAT+UNSAT Track* of the SAT competition 2014 [Bie14]. This improved version is even faster. We thank Armin Biere for providing the most recent version (as of 2015-06-05) of the lingeling SAT solver.

1. Table 2.K(a) shows that we have to pick bi_{Y}.
2. Tables 2.L show that the initial formulation for rotation systems of general degree is the best variant, so we pick ro_{R}.
3. There are eight possible parameter settings left. For this, we filter the data into a single Table 2.M. Here we see that with almost no difference in the number of solved instances, the variant with parameters in_{N}, cu_{Y} and sk_{Y} achieves the best run-time.

Thus, on this sample of 405 Rome graph instances we achieve the best result with binary face representations, our initial rotation system formulation, special variables and constraints for cubic vertices, as well as applying face skipping. We will use this parameter setting in the next paragraphs.

ILP and SAT Performance on Real World Graphs. Our C++ code is compiled with GCC 4.9.2, and runs on a single core of an AMD Opteron 6386 SE with DDR3 Memory @ 1600 MHz under Debian 8.0. We use the ILP solver CPLEX 12.6.1, the SAT solver lingeling, and the Open Graph Drawing Framework (www.ogdf.net, GPL), and apply a 72 GB memory limit.

We consider the established *Rome* and *North* benchmark sets of graphs collected from real-world applications. We use the ILP and SAT approaches to compute the genera of all 8249 (423) non-planar Rome (North) graphs. Each approach is run with a 30-minutes time limit for each graph to compute its genus; we omit 10 (*North*) instances that failed due to the memory limitation. Characteristics about the data sets and the resulting formulations can be found in Table 2.N.

Figure 2.O(a) shows the success rate (computations finished within the time limit) for the Rome graphs, depending on the number of vertices of the input graph. Both the SAT and ILP approach exhibit comparable numbers, but nearly always the success rate of the SAT approach is as good as or better than the ILP's. However, the differences are not significant. Instances with up to 40 vertices can be solved with a high success rate; our approach degrades heavily for graphs with more than 60–70 vertices. However, it is worth noting that even if the genus is not calculated to provable optimality, we obtain highly nontrivial bounds on the genus of the graphs in question.

In Figure 2.O(b) we see that, given *any* fixed time limit below 30 minutes, the SAT approach clearly solves more instances than the ILP approach. Note that the curve that corresponds to the solved SAT instances flattens out very quickly.

When we compare the success rates to the density of the NPC (see Figure 2.O(c)), we see the same characteristics as in Figure 2.O(a). Both approaches are able to solve instances with density (i.e., $|E|/|V|$) up to 1.6 with a high success rate but are typically not able to obtain provably optimal values for densities above 1.9.

Finally, we compare the average running times of the instances that are solved by both approaches. Out of the 8249 non-planar Rome graphs we are able to solve 2571 with SAT *and* ILP formulation, and additionally 96 (24) more with the SAT (ILP, respectively). Except for very small graphs, the average running time of the SAT approach is always at least one or two orders of magnitude lower than the average running time of the ILP approach, see Figure 2.O(d).

Considering the non-planar North graphs, Figure 2.O(e) shows that the success rates of both approaches are comparable again. As before, the differences are not significant. However, ten instances could not be solved due to the high memory consumption caused by the exponential number of constraints ($\mathbb{I}R_3^{\mathrm{I}}$) and rules ($\mathbb{B}R_5^{\mathrm{I}}$). Since the results for the North graphs are analogous to those for the Rome graphs, we omit discussing them in detail.

Generally, we observe that the SAT approach is particularly fast to show the existence of an embedding but relatively slow to prove that there is no embedding with a given number of faces.

(a) bin. face rep. config	solved bi_N	bi_Y	both	diff	avg. time[sec] bi_N	bi_Y	diff
cu_N in_N sk_N ro_R	108	150	108	42	33.2	1.8	−94%
cu_N in_N sk_N ro_B	104	135	103	31	118.8	5.1	−95%
cu_N in_N sk_N ro_I	102	145	102	43	113.9	22.5	−80%
cu_N in_N sk_Y ro_R	108	150	108	42	32.8	1.8	−94%
cu_N in_N sk_Y ro_B	104	135	103	31	119.2	5.1	−95%
cu_N in_N sk_Y ro_I	102	145	102	43	113.4	22.8	−79%
cu_N in_Y sk_N ro_R	112	151	112	39	104.0	1.3	−98%
cu_N in_Y sk_N ro_B	106	147	105	41	65.5	10.5	−83%
cu_N in_Y sk_N ro_I	98	146	98	48	49.5	10.4	−79%
cu_N in_Y sk_Y ro_R	110	147	110	37	60.9	1.6	−97%
cu_N in_Y sk_Y ro_B	105	146	105	41	75.4	1.0	−98%
cu_N in_Y sk_Y ro_I	101	141	101	40	88.8	15.6	−82%
cu_Y in_N sk_N ro_R	103	146	103	43	59.1	0.4	−99%
cu_Y in_N sk_N ro_B	103	142	103	39	124.1	3.9	−96%
cu_Y in_N sk_N ro_I	98	147	98	49	81.5	12.6	−84%
cu_Y in_N sk_Y ro_R	103	146	103	43	58.7	0.3	−99%
cu_Y in_N sk_Y ro_B	103	142	103	39	124.0	3.9	−96%
cu_Y in_N sk_Y ro_I	98	147	98	49	81.7	12.6	−84%
cu_Y in_Y sk_N ro_R	104	151	104	47	71.4	1.0	−98%
cu_Y in_Y sk_N ro_B	103	150	103	47	54.3	1.0	−98%
cu_Y in_Y sk_N ro_I	102	145	102	43	87.9	22.9	−73%
cu_Y in_Y sk_Y ro_R	105	150	105	45	52.7	0.9	−98%
cu_Y in_Y sk_Y ro_B	100	148	100	48	82.3	0.6	−99%
cu_Y in_Y sk_Y ro_I	102	144	102	42	89.2	5.3	−94%
average	103	146	103	43	80.8	6.8	−92%

(b) $d_v = 3$? config	solved cu_N	cu_Y	both	diff	avg. time[ms] cu_N	cu_Y	diff
bi_N in_N sk_N ro_R	108	103	100	-5	28.9	57.8	+99%
bi_N in_N sk_N ro_B	104	103	97	-1	98.2	81.5	−17%
bi_N in_N sk_N ro_I	102	98	95	-4	94.7	79.5	−15%
bi_N in_N sk_Y ro_R	108	103	100	-5	28.7	57.4	+100%
bi_N in_N sk_Y ro_B	104	103	97	-1	98.3	81.5	−17%
bi_N in_N sk_Y ro_I	102	98	95	-4	94.6	79.6	−15%
bi_N in_Y sk_N ro_R	112	104	101	-8	80.1	62.6	−21%
bi_N in_Y sk_N ro_B	106	103	100	-3	52.5	35.8	−31%
bi_N in_Y sk_N ro_I	98	102	95	4	40.3	62.0	+53%
bi_N in_Y sk_Y ro_R	110	105	99	-5	56.7	38.9	−31%
bi_N in_Y sk_Y ro_B	105	100	98	-5	56.1	66.6	+18%
bi_N in_Y sk_Y ro_I	101	102	96	1	79.8	57.4	−28%
bi_Y in_N sk_N ro_R	150	146	146	-4	8.7	5.3	−38%
bi_Y in_N sk_N ro_B	135	142	131	7	7.7	12.7	+65%
bi_Y in_N sk_N ro_I	145	147	143	2	27.6	19.1	−30%
bi_Y in_N sk_Y ro_R	150	146	146	-4	8.0	5.2	−34%
bi_Y in_N sk_Y ro_B	135	142	131	7	7.7	12.9	+67%
bi_Y in_N sk_Y ro_I	145	147	143	2	27.8	19.2	−31%
bi_Y in_Y sk_N ro_R	151	151	150	0	11.4	13.9	+22%
bi_Y in_Y sk_N ro_B	147	150	147	3	14.7	9.7	−33%
bi_Y in_Y sk_N ro_I	146	145	142	-1	26.5	30.1	+13%
bi_Y in_Y sk_Y ro_R	147	150	147	3	8.9	8.6	−3%
bi_Y in_Y sk_Y ro_B	146	148	145	2	6.1	7.2	+19%
bi_Y in_Y sk_Y ro_I	141	144	137	3	22.7	12.2	−46%
average	125	124	120	-1	36.1	33.4	−7%

(c) incr. form. config	solved in_N	in_Y	both	diff	avg. time[sec] in_N	in_Y	diff
bi_N cu_N sk_N ro_R	108	112	102	4	33.2	62.6	+88%
bi_N cu_N sk_N ro_B	104	106	101	2	118.7	52.7	−55%
bi_N cu_N sk_N ro_I	102	98	95	-4	63.9	38.5	−39%
bi_N cu_N sk_Y ro_R	108	110	101	2	26.4	35.6	+34%
bi_N cu_N sk_Y ro_B	104	105	101	1	116.9	47.8	−59%
bi_N cu_N sk_Y ro_I	102	101	95	-1	75.6	80.3	+6%
bi_N cu_Y sk_N ro_R	103	104	98	1	58.7	47.9	−18%
bi_N cu_Y sk_N ro_B	103	103	99	0	107.3	35.1	−67%
bi_N cu_Y sk_N ro_I	98	102	94	4	80.7	68.7	−14%
bi_N cu_Y sk_Y ro_R	103	105	98	2	53.2	45.5	−14%
bi_N cu_Y sk_Y ro_B	103	100	96	-3	93.9	52.5	−44%
bi_N cu_Y sk_Y ro_I	98	102	94	4	81.9	68.2	−16%
bi_Y cu_N sk_N ro_R	150	151	150	1	12.7	11.4	−10%
bi_Y cu_N sk_N ro_B	135	147	133	12	10.7	11.8	+9%
bi_Y cu_N sk_N ro_I	145	146	140	1	27.6	25.8	−6%
bi_Y cu_N sk_Y ro_R	150	147	147	-3	7.3	8.9	+22%
bi_Y cu_N sk_Y ro_B	135	146	134	11	10.6	5.5	−47%
bi_Y cu_N sk_Y ro_I	145	141	138	-4	28.5	22.6	−20%
bi_Y cu_Y sk_N ro_R	146	151	146	5	5.3	3.4	−36%
bi_Y cu_Y sk_N ro_B	142	150	142	8	14.4	3.8	−73%
bi_Y cu_Y sk_N ro_I	147	145	144	-2	22.6	30.2	+33%
bi_Y cu_Y sk_Y ro_R	146	150	146	4	5.2	4.6	−11%
bi_Y cu_Y sk_Y ro_B	142	148	141	6	14.2	4.5	−68%
bi_Y cu_Y sk_Y ro_I	147	144	142	-3	19.2	12.2	−36%
average	124	126	120	2	39.7	28.7	−28%

(d) face skip. config	solved sk_N	sk_Y	both	diff	avg. time[sec] sk_N	sk_Y	diff
bi_N cu_N in_N ro_R	108	108	108	0	33.2	32.8	−1%
bi_N cu_N in_N ro_B	104	104	104	0	122.6	122.8	±0%
bi_N cu_N in_N ro_I	102	102	102	0	113.9	113.4	±0%
bi_N cu_N in_Y ro_R	112	110	103	-2	77.3	51.9	−32%
bi_N cu_N in_Y ro_B	106	105	102	-1	54.4	53.9	±0%
bi_N cu_N in_Y ro_I	98	101	94	3	43.3	67.6	+56%
bi_N cu_Y in_N ro_R	103	103	103	0	59.1	58.7	±0%
bi_N cu_Y in_N ro_B	103	103	103	0	124.1	124.0	±0%
bi_N cu_Y in_N ro_I	98	98	98	0	81.5	81.7	±0%
bi_N cu_Y in_Y ro_R	104	105	97	1	48.5	43.3	−10%
bi_N cu_Y in_Y ro_B	103	100	96	-3	51.2	70.2	+37%
bi_N cu_Y in_Y ro_I	102	102	100	0	87.4	89.4	+2%
bi_Y cu_N in_N ro_R	150	150	150	0	12.7	12.2	−3%
bi_Y cu_N in_N ro_B	135	135	135	0	10.7	10.6	±0%
bi_Y cu_N in_N ro_I	145	145	145	0	27.9	28.1	±0%
bi_Y cu_N in_Y ro_R	151	147	147	-4	9.9	8.9	−10%
bi_Y cu_N in_Y ro_B	147	146	145	-1	13.7	6.3	−54%
bi_Y cu_N in_Y ro_I	146	141	138	-5	23.0	18.5	−19%
bi_Y cu_Y in_N ro_R	146	146	146	0	5.3	5.2	−1%
bi_Y cu_Y in_N ro_B	142	142	142	0	14.4	14.5	±0%
bi_Y cu_Y in_N ro_I	147	147	147	0	22.3	22.2	±0%
bi_Y cu_Y in_Y ro_R	151	150	150	-1	13.5	10.6	−21%
bi_Y cu_Y in_Y ro_B	150	148	148	-2	9.4	7.8	−17%
bi_Y cu_Y in_Y ro_I	145	144	140	-1	28.4	12.4	−56%
average	125	124	123	-1	40.2	38.9	−3%

2.K **Table 2.K:** Comparison of all 48 SAT variants. **(a)** effect of the binary face index reformulation, **(b)** effect of special variables and constraints for cubic vertices, **(c)** effect of the incremental formulation in the iterative process, **(d)** effect of face skipping in the iterative process. See Table 2.L for the remaining results.

(a) $\mathbb{B}^{\mathrm{I}}$ vs. $\mathbb{B}^{\mathrm{Bet}}$ config	solved $\mathrm{ro_R}$	$\mathrm{ro_B}$	both	diff	avg. time[sec] $\mathrm{ro_R}$	$\mathrm{ro_B}$	diff
$\mathrm{bi_N}$ $\mathrm{cu_N}$ $\mathrm{in_N}$ $\mathrm{sk_N}$	108	104	100	-4	26.8	97.5	+264%
$\mathrm{bi_N}$ $\mathrm{cu_N}$ $\mathrm{in_N}$ $\mathrm{sk_Y}$	108	104	100	-4	26.5	97.4	+267%
$\mathrm{bi_N}$ $\mathrm{cu_N}$ $\mathrm{in_Y}$ $\mathrm{sk_N}$	112	106	103	-6	50.7	51.3	+1%
$\mathrm{bi_N}$ $\mathrm{cu_N}$ $\mathrm{in_Y}$ $\mathrm{sk_Y}$	110	105	101	-5	51.3	63.5	+23%
$\mathrm{bi_N}$ $\mathrm{cu_Y}$ $\mathrm{in_N}$ $\mathrm{sk_N}$	103	103	96	0	57.5	81.1	+40%
$\mathrm{bi_N}$ $\mathrm{cu_Y}$ $\mathrm{in_N}$ $\mathrm{sk_Y}$	103	103	96	0	57.0	80.6	+41%
$\mathrm{bi_N}$ $\mathrm{cu_Y}$ $\mathrm{in_Y}$ $\mathrm{sk_N}$	104	103	96	-1	57.1	32.7	−42%
$\mathrm{bi_N}$ $\mathrm{cu_Y}$ $\mathrm{in_Y}$ $\mathrm{sk_Y}$	105	100	96	-5	48.5	56.6	+16%
$\mathrm{bi_Y}$ $\mathrm{cu_N}$ $\mathrm{in_N}$ $\mathrm{sk_N}$	150	135	135	-15	6.2	10.7	+71%
$\mathrm{bi_Y}$ $\mathrm{cu_N}$ $\mathrm{in_N}$ $\mathrm{sk_Y}$	150	135	135	-15	6.2	10.6	+71%
$\mathrm{bi_Y}$ $\mathrm{cu_N}$ $\mathrm{in_Y}$ $\mathrm{sk_N}$	151	147	146	-4	10.9	14.2	+28%
$\mathrm{bi_Y}$ $\mathrm{cu_N}$ $\mathrm{in_Y}$ $\mathrm{sk_Y}$	147	146	144	-1	6.3	5.9	−5%
$\mathrm{bi_Y}$ $\mathrm{cu_Y}$ $\mathrm{in_N}$ $\mathrm{sk_N}$	146	142	142	-4	4.5	14.4	+217%
$\mathrm{bi_Y}$ $\mathrm{cu_Y}$ $\mathrm{in_N}$ $\mathrm{sk_Y}$	146	142	142	-4	4.5	14.5	+224%
$\mathrm{bi_Y}$ $\mathrm{cu_Y}$ $\mathrm{in_Y}$ $\mathrm{sk_N}$	151	150	150	-1	14.2	10.1	−29%
$\mathrm{bi_Y}$ $\mathrm{cu_Y}$ $\mathrm{in_Y}$ $\mathrm{sk_Y}$	150	148	148	-2	10.7	7.8	−26%
average	128	123	121	-5	23.9	35.2	+47%

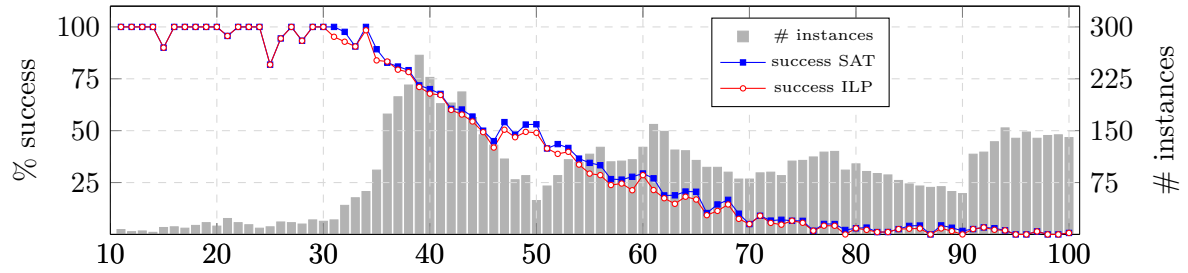
(b) $\mathbb{B}^{\mathrm{I}}$ vs. $\mathbb{B}^{\mathrm{Ind}}$ config	solved $\mathrm{ro_R}$	$\mathrm{ro_I}$	both	diff	avg. time[sec] $\mathrm{ro_R}$	$\mathrm{ro_I}$	diff
$\mathrm{bi_N}$ $\mathrm{cu_N}$ $\mathrm{in_N}$ $\mathrm{sk_N}$	108	102	100	-6	27.3	97.6	+256%
$\mathrm{bi_N}$ $\mathrm{cu_N}$ $\mathrm{in_N}$ $\mathrm{sk_Y}$	108	102	100	-6	27.1	97.3	+259%
$\mathrm{bi_N}$ $\mathrm{cu_N}$ $\mathrm{in_Y}$ $\mathrm{sk_N}$	112	98	98	-14	51.2	49.5	−3%
$\mathrm{bi_N}$ $\mathrm{cu_N}$ $\mathrm{in_Y}$ $\mathrm{sk_Y}$	110	101	97	-9	30.8	73.7	+139%
$\mathrm{bi_N}$ $\mathrm{cu_Y}$ $\mathrm{in_N}$ $\mathrm{sk_N}$	103	98	94	-5	57.3	50.6	−11%
$\mathrm{bi_N}$ $\mathrm{cu_Y}$ $\mathrm{in_N}$ $\mathrm{sk_Y}$	103	98	94	-5	56.9	50.6	−11%
$\mathrm{bi_N}$ $\mathrm{cu_Y}$ $\mathrm{in_Y}$ $\mathrm{sk_N}$	104	102	97	-2	61.2	65.4	+6%
$\mathrm{bi_N}$ $\mathrm{cu_Y}$ $\mathrm{in_Y}$ $\mathrm{sk_Y}$	105	102	96	-3	46.1	78.7	+70%
$\mathrm{bi_Y}$ $\mathrm{cu_N}$ $\mathrm{in_N}$ $\mathrm{sk_N}$	150	145	145	-5	7.4	27.9	+275%
$\mathrm{bi_Y}$ $\mathrm{cu_N}$ $\mathrm{in_N}$ $\mathrm{sk_Y}$	150	145	145	-5	6.8	28.1	+313%
$\mathrm{bi_Y}$ $\mathrm{cu_N}$ $\mathrm{in_Y}$ $\mathrm{sk_N}$	151	146	145	-5	10.9	36.4	+231%
$\mathrm{bi_Y}$ $\mathrm{cu_N}$ $\mathrm{in_Y}$ $\mathrm{sk_Y}$	147	141	139	-6	6.0	21.8	+261%
$\mathrm{bi_Y}$ $\mathrm{cu_Y}$ $\mathrm{in_N}$ $\mathrm{sk_N}$	146	147	145	1	5.3	20.1	+278%
$\mathrm{bi_Y}$ $\mathrm{cu_Y}$ $\mathrm{in_N}$ $\mathrm{sk_Y}$	146	147	145	1	5.2	20.1	+284%
$\mathrm{bi_Y}$ $\mathrm{cu_Y}$ $\mathrm{in_Y}$ $\mathrm{sk_N}$	151	145	145	-6	12.8	30.9	+141%
$\mathrm{bi_Y}$ $\mathrm{cu_Y}$ $\mathrm{in_Y}$ $\mathrm{sk_Y}$	150	144	143	-6	2.5	12.8	+419%
average	128	123	120	-5	22.2	43.3	+95%

(c) $\mathbb{B}^{\mathrm{Ind}}$ vs. $\mathbb{B}^{\mathrm{Bet}}$ config	solved $\mathrm{ro_I}$	$\mathrm{ro_B}$	both	diff	avg. time[sec] $\mathrm{ro_I}$	$\mathrm{ro_B}$	diff
$\mathrm{bi_N}$ $\mathrm{cu_N}$ $\mathrm{in_N}$ $\mathrm{sk_N}$	102	104	99	2	107.8	107.0	±0%
$\mathrm{bi_N}$ $\mathrm{cu_N}$ $\mathrm{in_N}$ $\mathrm{sk_Y}$	102	104	99	2	107.5	107.3	±0%
$\mathrm{bi_N}$ $\mathrm{cu_N}$ $\mathrm{in_Y}$ $\mathrm{sk_N}$	98	106	96	8	39.5	39.4	±0%
$\mathrm{bi_N}$ $\mathrm{cu_N}$ $\mathrm{in_Y}$ $\mathrm{sk_Y}$	101	105	98	4	83.7	51.6	−38%
$\mathrm{bi_N}$ $\mathrm{cu_Y}$ $\mathrm{in_N}$ $\mathrm{sk_N}$	98	103	96	5	81.4	74.3	−8%
$\mathrm{bi_N}$ $\mathrm{cu_Y}$ $\mathrm{in_N}$ $\mathrm{sk_Y}$	98	103	96	5	81.6	74.4	−8%
$\mathrm{bi_N}$ $\mathrm{cu_Y}$ $\mathrm{in_Y}$ $\mathrm{sk_N}$	102	103	98	1	72.4	38.1	−47%
$\mathrm{bi_N}$ $\mathrm{cu_Y}$ $\mathrm{in_Y}$ $\mathrm{sk_Y}$	102	100	96	-2	69.5	61.6	−11%
$\mathrm{bi_Y}$ $\mathrm{cu_N}$ $\mathrm{in_N}$ $\mathrm{sk_N}$	145	135	134	-10	24.4	10.2	−58%
$\mathrm{bi_Y}$ $\mathrm{cu_N}$ $\mathrm{in_N}$ $\mathrm{sk_Y}$	145	135	134	-10	24.6	10.1	−58%
$\mathrm{bi_Y}$ $\mathrm{cu_N}$ $\mathrm{in_Y}$ $\mathrm{sk_N}$	146	147	143	1	34.4	14.9	−56%
$\mathrm{bi_Y}$ $\mathrm{cu_N}$ $\mathrm{in_Y}$ $\mathrm{sk_Y}$	141	146	137	5	19.4	5.8	−69%
$\mathrm{bi_Y}$ $\mathrm{cu_Y}$ $\mathrm{in_N}$ $\mathrm{sk_N}$	147	142	141	-5	18.5	12.0	−34%
$\mathrm{bi_Y}$ $\mathrm{cu_Y}$ $\mathrm{in_N}$ $\mathrm{sk_Y}$	147	142	141	-5	18.4	12.1	−34%
$\mathrm{bi_Y}$ $\mathrm{cu_Y}$ $\mathrm{in_Y}$ $\mathrm{sk_N}$	145	150	145	5	30.9	9.1	−70%
$\mathrm{bi_Y}$ $\mathrm{cu_Y}$ $\mathrm{in_Y}$ $\mathrm{sk_Y}$	144	148	141	4	11.8	4.2	−64%
average	122	123	118	1	46.6	34.3	−26%

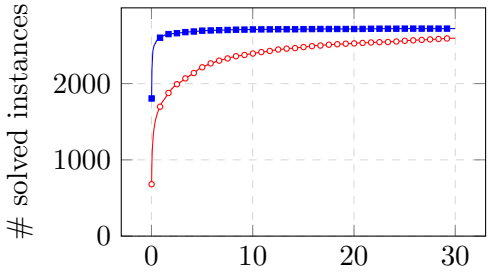
Table 2.L: Comparison of all 48 SAT variants. **(a)** initial formulation vs. betweenness reformulation, **(b)** initial formulation vs. index reformulation, **(c)** index reformulation vs. betweenness reformulation. See Table 2.K for the remaining results. 2.L

	$\mathrm{in_Y}$ $\mathrm{cu_Y}$ $\mathrm{sk_Y}$	$\mathrm{in_Y}$ $\mathrm{cu_Y}$ $\mathrm{sk_N}$	$\mathrm{in_Y}$ $\mathrm{cu_N}$ $\mathrm{sk_Y}$	$\mathrm{in_Y}$ $\mathrm{cu_N}$ $\mathrm{sk_N}$	$\mathrm{in_N}$ $\mathrm{cu_Y}$ $\mathrm{sk_Y}$	$\mathrm{in_N}$ $\mathrm{cu_Y}$ $\mathrm{sk_N}$	$\mathrm{in_N}$ $\mathrm{cu_N}$ $\mathrm{sk_Y}$	$\mathrm{in_N}$ $\mathrm{cu_N}$ $\mathrm{sk_N}$
# solved instances	150	151	147	151	146	150	146	150
avg. run-time [sec]	8.6	13.9	8.9	11.4	5.2	8.7	5.3	8.0

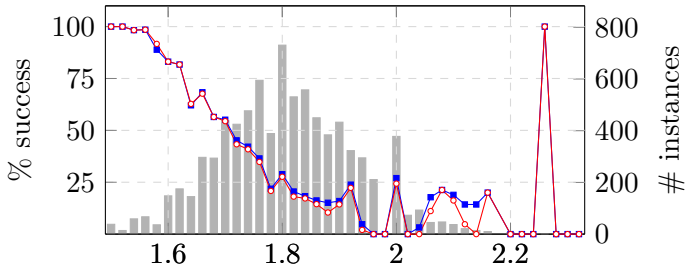
Table 2.M: Compressed version of the results of Table 2.K and 2.L when we restrict the data to $\mathrm{ro_R}$ and $\mathrm{bi_Y}$. 2.M



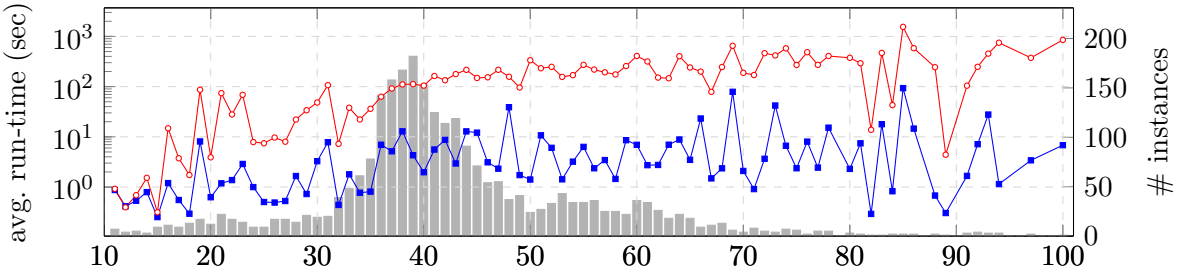
(a) input graph $|V|$ (Rome)



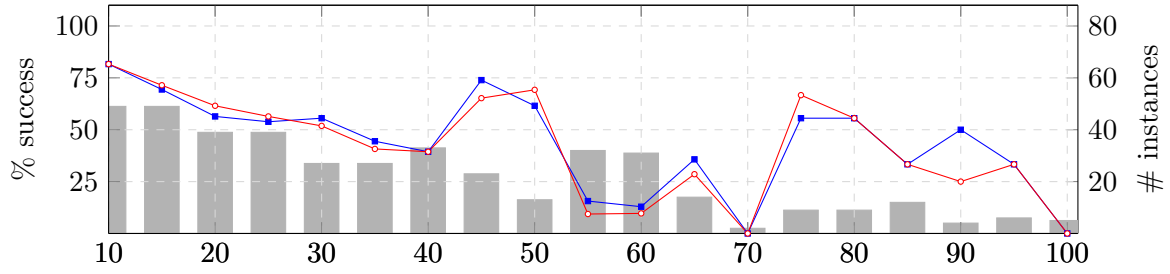
(b) time in min. (Rome)



(c) NPC density (Rome)



(d) input graph $|V|$ (Rome)



(e) input graph $|V|$ (North)

2.O **Figure 2.O:** Rome Graphs: **(a)** success rate per number of nodes, **(b)** solved instances per given time, **(c)** success rate per non-planar core density $|E|/|V|$, **(d)** average running time per number of nodes where both approaches were successful. North graphs: **(e)** success rate per number of nodes.

Rome						*North*					
range	average for computation on NPC					range	average for computation on NPC				
$\|V\|$	$\|V\|$	$\%\|V_3\|$	$\bar{f}$	#vars	#cons	$\|V\|$	$\|V\|$	$\%\|V_3\|$	$\bar{f}$	#vars	#cons
10–40	12.8	64.2	10.0	616.1	3399.5	10–40	12.6	38.3	17.4	2200.0	102295.9
41–60	18.5	60.3	15.3	1310.7	7639.9	41–60	24.6	40.3	29.9	4916.7	197577.3
61–80	26.8	59.4	22.5	2624.4	15735.1	61–80	32.1	43.5	35.5	7741.7	249864.6
81–100	36.4	58.5	30.9	4718.4	28778.3	81–100	24.3	40.6	34.7	7146.7	632634.6

Table 2.N: Characteristics of instances and resulting formulations. The graphs from the *Rome* (left table) and *North* (right table) benchmark sets are grouped by their number of vertices in the given ranges. For each group, we give the averages for the following values: number of vertices and percentage of degree-three vertices in the NPC, upper bound $\bar{f}$ on the number of faces, number of variables and constraints in the ILP formulation. 2.N

This is of particular interest for non-planar graphs that allow a genus-1 embedding, since there the SAT formulation is quick to find such a solution and need not prove that a lower surface is infeasible. The SAT formulation's behavior in fact suggests an easy heuristical approach: if solving the SAT instance for f faces needs a disproportionally long running time (compared to the previous iterations for lower face numbers), this typically indicates that it is an unsatisfiable instance and $f-2$ faces is the optimal value.

Comparison to Existing Genus Computations. An evaluation of exhaustive search algorithms for determining the genus distribution of complete graphs was performed in [Sch12]. Fixing the rotation of the first vertex, it is possible to compute the genus distribution of the complete graph K_7 within 896 hours of computation (112 hours on 8 parallel threads). While *both* our approaches perform significantly better, there is a notable (and w.r.t. to the above evaluations particularly surprising) difference in their performance: the SAT approach needs 1 hour to find and prove the optimal genus; solving the ILP takes only 30 seconds.

A circulant $C_n(S)$ is the Cayley graph of $\mathbb{Z}_n$ with generating set S. Conder and Grande [CG15] recently characterized all circulants with genus 1 and 2. A crucial part of the characterization is the determination of the genus of several sporadic cases where the lower bounds are more problematic. At the same time, these sporadic cases constitute the main obstacle in both obtaining a simpler proof, as well as extending the results to higher genera. By far the most difficult case is proving that the genus of $C_{11}(1,2,4)$ is at least 3. The proof takes three pages of theoretical analysis and eventually resorts to a computational verification of three subcases, taking altogether around 85 hours using the MAGMA computational algebra system in a nontrivial problem-specific setting. The ILP solver needs 180 hours (7.5 days) to determine the genus without using any theoretical results or problem-specific information. The SAT solver takes roughly 2 weeks.

2.8 Minimum Genus on Non-Orientable Surfaces

Remember the definition of facial walks on a non-orientable surface. By σ we denote the current state (initialized with $\sigma = 1$). After the traversal of an edge $e = vu$ we update the state $\sigma := \lambda(e)\sigma$ and continue along $\pi_u^\sigma(e)$.

Determining the minimum genus is now equivalent to finding a rotation system π together with edge signs λ such that the number of faces is maximized over all feasible (λ, π) pairs.

Recall the definitions in Lemma 1.3. By $S(e) := S_{\mathrm{N}}(e)$ we denote the set of both directions of each edge e with the possible facial walk states that we had before traversing the edge e itself.

Let $\mathcal{S} := \bigcup_{e\in E} S(e)$. In the orientable case we ensured that we traverse each edge exactly twice (each of the two darts exactly once). We now have to guarantee that for each edge e exactly two of the states in $S(e)$ are used (cf. Lemma 1.3). Thus, we introduce variables z_s for $s \in \mathcal{S}$ to denote if s is used in one of the faces.

Furthermore, we use the new variable ℓ_e for each edge which is `true` if and only if $\lambda(e) = 1$.

The old containment variables c_a^i are now replaced by c_s^i variables (for $i \in [f]$ and $s \in \mathcal{S}$). We continue to use the $p_{u,w}^v$ variables to denote that $vw = \pi_v(uv)$. Note that π^{-1} (needed if the current state is $\sigma = -1$) is represented by swapping the indices of the p variables. We keep the constraints $\mathbb{B}R^{\mathrm{I}}$ to force them into a rotation system.

The remaining part of the formulation $\mathbb{B}^{\mathrm{NonO}}$ for the non-orientable case is:

$$\neg(c_s^i \wedge c_s^j) \qquad \forall s \in \mathcal{S},\, i < j \in [f] \qquad (\mathbb{B}C_1^{\mathrm{NonO}})$$

$$\bigvee\nolimits_{s\in\mathcal{S}} c_s^i \qquad \forall i \in [f] \qquad (\mathbb{B}C_2^{\mathrm{NonO}})$$

$$\bigvee\nolimits_{i\in[f]} c_s^i \;\leftrightarrow\; z_s \qquad \forall s \in \mathcal{S} \qquad (\mathbb{B}C_3^{\mathrm{NonO}})$$

$$\bigvee\nolimits_{S\subset S(e),|S|=2} \bigwedge\nolimits_{s\in S} z_s \qquad \forall e \in E \qquad (\mathbb{B}C_4^{\mathrm{NonO}})$$

$$\neg\Big(\bigvee\nolimits_{S\subset S(e),|S|=3} \bigwedge\nolimits_{s\in S} z_s\Big) \qquad \forall e \in E \qquad (\mathbb{B}C_5^{\mathrm{NonO}})$$

$$p_{u,w}^v \wedge c_{uv,+}^i \wedge \ell_{\{v,w\}} \;\rightarrow\; c_{vw,+}^i \qquad \forall v \in V,\, u \neq w \in N(v),\, i \in [f] \qquad (\mathbb{B}W_1^{\mathrm{NonO}})$$

$$p_{u,w}^v \wedge c_{uv,+}^i \wedge \neg\ell_{\{v,w\}} \;\rightarrow\; c_{vw,-}^i \qquad \forall v \in V,\, u \neq w \in N(v),\, i \in [f] \qquad (\mathbb{B}W_2^{\mathrm{NonO}})$$

$$p_{w,u}^v \wedge c_{uv,-}^i \wedge \ell_{\{v,w\}} \;\rightarrow\; c_{vw,-}^i \qquad \forall v \in V,\, u \neq w \in N(v),\, i \in [f] \qquad (\mathbb{B}W_3^{\mathrm{NonO}})$$

$$p_{w,u}^v \wedge c_{uv,-}^i \wedge \neg\ell_{\{v,w\}} \;\rightarrow\; c_{vw,+}^i \qquad \forall v \in V,\, u \neq w \in N(v),\, i \in [f] \qquad (\mathbb{B}W_4^{\mathrm{NonO}})$$

2.3a
$$\bigvee\nolimits_{e\in E} \neg\ell_e \qquad (2.3a)$$

Theorem. *The SAT formulation $\mathbb{B}^{\mathrm{NonO}}$ above solves MGP on non-orientable surfaces with respect to f.*

Proof. As before, the $\mathbb{B}R^{\mathrm{I}}$ constraints ensure that we have a rotation system π (induced by the p^v variables). The signature assignment λ is arbitrary and given by the ℓ_e variables, but we ensure that we have an embedding on a non-orientable surface (see Lemma 1.2) by forcing at least one of the ℓ_e to be `false`, see (2.3a). Otherwise, we would have all signs $+1$, which is an embedding on an oriented surface. In $(\mathbb{B}W_1^{\mathrm{NonO}})$–$(\mathbb{B}W_4^{\mathrm{NonO}})$ we have all four cases that can occur when traversing an edge (the state when going into v by uv is $+/-$, and the state of the successor/predecessor in the rotation has sign $+/-$). The constraints $(\mathbb{B}C_4^{\mathrm{NonO}})$ and $(\mathbb{B}C_5^{\mathrm{NonO}})$ select exactly two elements of $S(e)$ for each edge e. By $(\mathbb{B}C_3^{\mathrm{NonO}})$ we assign a face index to each used state $s \in \mathcal{S}$. Additionally, $(\mathbb{B}C_2^{\mathrm{NonO}})$ prevents faces from being empty. Finally, $(\mathbb{B}C_1^{\mathrm{NonO}})$ ensures that a single state is not in two (or more) faces simultaneously. □

Note that (2.3a) does not fit into our decomposition of the formulations in six blocks.

Remark. All of the speed-up techniques and the transformations for polynomially sized formulations can be applied in the non-orientable case. ILP models are analogous to the SAT formulation described above. ⌟

We tested our model on graphs where we have a known minimum non-orientable genus greater than zero. Examples are

- K_5, see [Whi84, Theorem 11.19];

- $K_{3,3}$, see [Rin65b, Equation (1)];
- the subgroup graphs[6] of the groups $\mathbb{Z}_4 \rtimes \mathbb{Z}_4$, $\mathbb{Z}_9 \rtimes \mathbb{Z}_9$ and the dihedral group D_{16}. This result can be found in [MBS12] which is a generalization of the underlying work [BR06].

We did not perform any detailed run-time experiments in the non-orientable case. The computations for the K_5 and $K_{3,3}$ took a few days. The remaining examples required a few weeks each. We were able to achieve a significant speed-up using the following result.

Definition (local change, equivalent embeddings). A *local change* of an embedding (λ, π) changes the clockwise ordering to an anti-clockwise ordering at some vertex v, *i.e.*, π_v is replaced by its inverse π_v^{-1}, and $\lambda(e)$ is replaced by $-\lambda(e)$ for all edges e that are incident with v. Two embeddings are *equivalent* if one can be obtained from the other by a sequence of local changes. ⌟

Lemma 2.4. [MT01, p. 100] *Let $\Pi = (\lambda, \pi)$ be an embedding. For an arbitrary spanning tree T there is an embedding equivalent to Π such that the signs of the edges in T are all positive.* 2.4

Using the lemma above, we can strengthen constraint (2.3a) to

$$\bigwedge\nolimits_{e \in T} \ell_e \wedge \Big(\bigvee\nolimits_{e \notin T} \neg \ell_e \Big), \qquad (2.3a') \quad 2.3a'$$

where we fixed an arbitrary spanning tree T of G.

2.9 Conclusion and Open Problems

The MGP is very difficult from the mathematical, algorithmic, and practical perspective—the problem space is large and seems not to be well-structured, the existing algorithms are error-prone and/or very difficult to implement, and only little progress was made on the (practice-oriented) algorithmic side. In this chapter we have presented the first ILP and SAT formulations, together with several variants and alternative reformulations, for the problem, and investigated them in an experimental study. Our approach leads to the first (even easily!) implementable general-purpose minimum genus algorithms. Besides yielding practical algorithms for small to medium-sized graphs and small genus, one of the further advantages of our approach is that the formulations are adaptable and can be modified to tackle related problems. For example, the existence of polyhedral embeddings [MT01], or embeddings with given face lengths, say 5 and 6 as in the case of graph-theoretic models of carbon molecules, so-called fullerenes, see [Dez$^+$00].

On the negative side, our implementations cannot deal with too large graphs without resorting to extensive computational resources. However, this is not very surprising considering the difficulty of the problem—a fast exact algorithm could be used to solve several long-standing open problems, such as completing the list of forbidden toroidal minors. We also see—and hope for—certain similarities to the progress on exact algorithms for the well-known crossing number problem: while the first published report [Buc$^+$05] was only capable of solving Rome graphs with 30–40 vertices, it led to a series of improvements that culminated in the currently strongest variant [CMB08] which is capable of tackling even the largest Rome graphs.

There are several open questions and unused ideas to further strengthen the formulations or find the ideal trade-off between run-time and memory usage. Some experimental work will be necessary to find the optimal threshold ϑ (defined in Section 2.4) where we switch from exponentially (in the constant parameter ϑ) sized formulation for rotation systems to one of

[6] [BR06, Def. 1.1] The *subgroup graph* of a group is the graph whose vertices are the subgroups of the group and two vertices, say H_1 and H_2, are connected by an edge if and only if $H_1 \leq H_2$ and there is no subgroup K such that $H_1 \lneq K \lneq H_2$.

the presented polynomially sized formulations. The same idea can be used to investigate the effect of special variables and constraints for low-degree vertices. Here we only considered the cubic ($d_v = 3$) case. The principle also works for higher degrees and it could be worthwhile to spend the time searching for an algorithmic formulation to create the necessary constraints in a parameterized approach. This could even be used to omit the p^v variables for general degree completely and create specialized variables for each vertex degree.

Recently, there was a result [CR17][7] by Chen and Reidys about a new "easy-to-check" necessary condition for a given embedding to be an embedding of minimum genus. It is based on an interesting combinatorial way to represent graphs: Let π be a permutation. By $|\pi|$ we denote the number of cycles in π. A *fatgraph* is a triple (α, β, δ) of permutations on $[2m]$ where α is a fixed-point free involution (*i.e.*, $\alpha^2 = 1$) and $\delta = \alpha\beta$. The connection to our graphs is then: We label the dart set A of an input graph G using labels from the set $[2|E|]$ so that each label appears exactly once. This induces two permutations α and β, where α is a fixed-point free involution, whose cycles consist of the labels of the two darts of the same edge and β-cycles represent the counter-clockwise cyclic arrangement of all darts incident to the same vertex. It follows that

$$|\beta| - |\alpha| + |\delta| = 2 - 2g.$$

The necessary criterion for an embedding to be a minimum genus embedding is then given by [CR16, Corollary 4.8].

As mentioned earlier, we did not use PBS solvers for our formulations. It would be interesting to see how they compare to our initial formulation as well as how our optimized formulations (including face skipping) compare to simple PBS formulations.

Finally, we briefly sketched how to build formulations for the MGP on non-orientable surfaces. The methods of this chapter can be applied to such formulations. But furthermore, there are additional results to strengthen the according formulations, for example:

Lemma. [MT01, Lem. 4.1.3] *If W is a (λ, π)-facial walk, then the number of appearances of edges on W with negative signature is even.*

Considering non-orientable surfaces, we face the question if there are suitable benchmark instances to test algorithmic minimum genus computations. The *Rome* and *North* instances are natural candidates. Also the classification of circulants is a source for test instances. Due to the increased complexity by also considering edge signs we expect that our methods are only able solve MGP for comparably small instances.

[7] See [CR16] for a publicly available version.

Chapter 3

Limits of Greedy Approximation Algorithms for the Maximum Planar Subgraph Problem

In this chapter we consider approximation algorithms for the Maximum Planar Subgraph problem (MPS). The Maximum Planar Subgraph problem asks for a planar subgraph with maximum edge cardinality of a given simple, undirected graph. It is known to be MaxSNP-hard [Căl+98, Theorem 4.1] and the currently best known approximation algorithm achieves a ratio of 4/9.

We analyze the general limits of approximation algorithms for MPS, based either on planarity tests or on greedy inclusion of certain subgraphs. On the one hand, we cover upper bounds for the approximation ratios. On the other hand, we show NP-hardness for thereby arising subproblems, which hence would have to be approximated themselves. We also provide simpler proofs for already known facts.

In addition to the original publication [CHW16], this chapter includes notes on worst-case examples with non-constant skewness for the best currently known approximation algorithm by Călinescu et al. [Căl+98], on algorithms based on graph decompositions (cut- and path-width) and an alternative proof for the 7/18-approximation algorithm in [Căl+98].

3.1 Introduction

The aim of our work in the field of MPS approximation algorithms was to study the approximation ratio of variants of the currently best known algorithm "*Cactus Algorithm*" [Căl+98]. We also wanted to investigate new algorithms with better ratios. As our work did not result in such algorithms, we focused on limits for the ratio of MPS approximation algorithms. The insights we gained from studying special algorithms allow us to bound the approximation guarantee for several classes of approximation algorithms. The focus of our work is on:

- Algorithms inspired by planarity tests. Planarity test algorithms can easily be extended to obtain MPS heuristics. We show that such algorithms can build structures during the planarity test that likely prohibit the expansion to good MPS approximations.
- Algorithms inspired by generalized versions of the Cactus Algorithm. The Cactus Algorithm searches for triangles sharing at most one vertex with each other and connecting the thereby build cactus structures with single edges. Greedy algorithms that search for denser structures are a natural generalization of this algorithm. We show that there are limits on the approximation ratio of such generalized versions.

Recently, Chalermsook and Schmid [CS17] found a greedy algorithm that searches for structures denser than triangles. Their algorithm is the now best greedy approximation algorithm

for MPS but it still does not achieve the ratio of the algorithm by [Căl+98]. Note that no examples for the tightness of the algorithm by Chalermsook and Schmid are known, only a lower bound on the approximation ratio. Our upper bound from Theorem 3.17 applies also to this generalization variant of the Cactus Algorithm.

3.2 Maximality

While the Observation 3.1 is already known to be true, we provide a simpler instance than the original source [DFF85] by Dyer et al. They use a 3-colorable planar triangulated graph extended by $\Theta(n)$ edges that form three cycles on the node partitions induced by the coloring. Our argument is based on a K_5.

Definition (bundle). For two nodes u and v, we define a *u-v-bundle $\mathcal{B}^t_{u,v}$ of thickness t* as a set of t parallel 2-paths between u and v; the new inner nodes $\mathcal{I}(\mathcal{B}^t_{u,v})$ have degree 2. ⌟

3.1 **Observation 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 $P = p_1, \ldots, p_\ell$ on the nodes $\mathcal{I}(\mathcal{B}^\ell_{v,u})$. Let

$$S := \big(E(K_5) \setminus \{vu\}\big) \cup \{vp_k \mid k \text{ odd}\} \cup \{p_k u \mid k \text{ even}\}.$$

S is a maximal planar subgraph of G since adding any edge yields a K_5 subdivision (cf. Figure 3.A). An MPS H can be obtained from G by removing any one edge outside of $\mathcal{B}^\ell_{v,u}$. The approximation ratio is thus at most

$$\lim_{\ell\to\infty} \frac{|S|}{|E(H)|} = \lim_{\ell\to\infty} \frac{|E| - 1 + \ell}{|E| - 1 + 3\ell} = \frac{1}{3}.$$ □

3.3 Algorithms Inspired by Planarity Tests

First, we focus on DFS- and 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 planarity-testing in fact solves MPS-DFS heuristically.

Definition (k-book, k-book embedding). 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. ⌟

Definition (circle graph, overlap graph). 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. ⌟

Definition (c-CIG). 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). ⌟

Lemma. [CL91] *c-CIG is NP-hard for $c \geq 2$.*

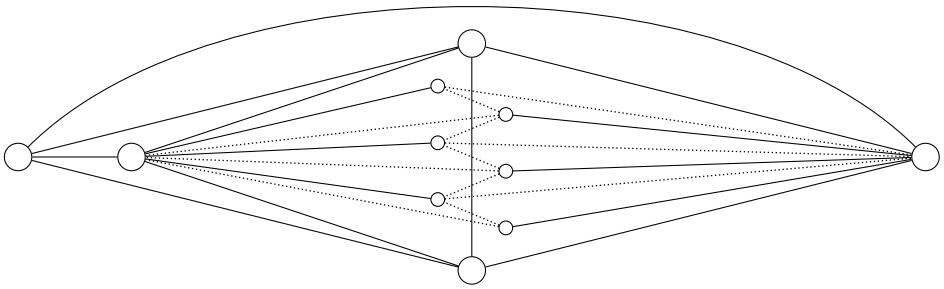


Figure 3.A: (cf. Obs. 3.1) Maximal planar subgraph S (stroked edges) for $\ell = 6$. 3.A

We will use the following result combined with the lemma above to show that MPS-DFS is hard.

Lemma. *Any solution for c-CIG corresponds to a c-book embedding of the respective overlap graph.*

Proof. The circle corresponds to the spine and each color class is embedded in its own page. □

Theorem 3.2. *MPS-DFS is NP-hard.* 3.2

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 $n := |W|$, $m := |F|$, and $\pi\colon [n] \to W$ denote the cyclic order of W induced by C. Let $B_i := \mathcal{B}^m_{\pi_i,\pi_{i+1}}$ denote a π_i-π_{i+1}-bundle of thickness m and $B'_i := B_i \cup \{\pi_i\pi_{i+1}\}$. We construct the input graph $D := \left(\bigcup_{i\in[n]} V(B_i), F \cup E_\mathrm{B}\right)$ for MPS-DFS, with $E_\mathrm{B} := \bigcup_{i\in[n]} E(B'_i)$; see Figure 3.B. The set

$$T := \{\pi_i\pi_{i+1}, u\pi_{i+1} \mid 0 \le i < n-1, u \in \mathcal{I}(B_i)\} \cup \{u\pi_{n-1} \mid u \in \mathcal{I}(B_{n-1})\},$$

is a DFS tree of D: 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 $\mathcal{I}(B_{n-1})$.

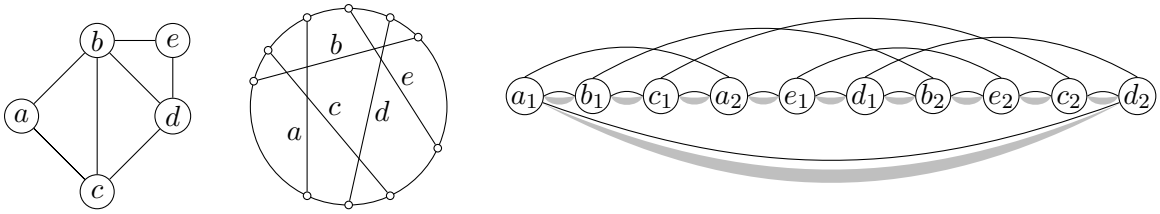
We show that the 2-CIG instance (G, k) has a solution if and only if D has a planar subgraph of size $\xi := k + n(2m+1)$ that contains T.

(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 $\pi_0, \pi_1, \ldots, \pi_{n-1}, \pi_0$, 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 $|E(S)| - |E_\mathrm{B}| = k$.

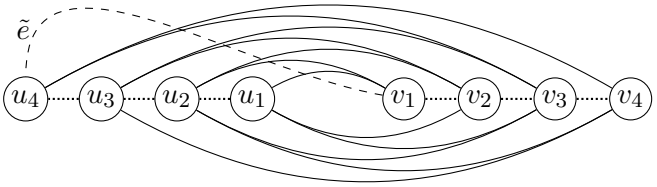
(Only If) Assume there is a solution for 2-CIG on (G, k). This corresponds to a 2-book embedding of a subgraph $C' := (W, E')$ of C, where the vertices W are placed on the spine according to π, and $|E'| \ge k$. Adding E_B to C' yields a planar graph. Note that $T \subseteq E_\mathrm{B}$ and C' contains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

Corollary. *There are (infinitely many) graphs G that allow a DFS tree T_v for each possible start node v such that MPS-DFS on each (G, T_v) is NP-hard.*

Proof. The proof for Theorem 3.2 works independently of the DFS start node since π is cyclic and π_0 can be chosen arbitrarily. □



3.B **Figure 3.B:** (cf. Theorem 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 $\pi_0 = a_1, \pi_1 = b_1, \ldots$ on the right (bundles sketched in gray).



3.C **Figure 3.C:** (cf. Theorem 3.3) Drawing of the graph G with $p = 4$. Edges of S are dotted.

We will see that any algorithm adding edges to an arbitrary DFS tree has an approximation ratio of at most 2/3. However, the corollary 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.

3.3 **Theorem 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

$$G := (V, S \cup \{\tilde{e}\} \cup T) \quad \text{with} \quad V := \{u_1, \ldots, u_p, v_1, \ldots, v_p\} \quad \text{and} \quad S := \bigcup\nolimits_{i=1}^{p-1} \{u_i u_{i+1}, v_i v_{i+1}\}.$$

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. Figure 3.C) in a way such that

3.4
$$\forall e \in T\colon e = u_i v_j \wedge |i - j| \leq 2. \tag{3.4}$$

Finally, we define $\tilde{e} := u_p v_1$. Observe that $|T| = 4p - 4$ and $P := S \cup \{\tilde{e}\}$ 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 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 $G[P]$ 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 Equation (3.4) 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. We conclude that MPS-DFS approximates MPS by a ratio of at most

$$\lim_{p \to \infty} \frac{|P| + |T| - (2p - 10)}{|E(G) \setminus \{\tilde{e}\}|} = \frac{2}{3}. \qquad \square$$

We wonder if this result is caused by the special structure of DFS trees or if this can be extended, for example, to BFS-based algorithms:

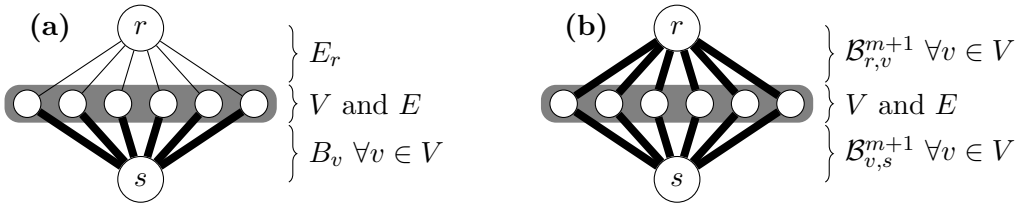


Figure 3.D: (a) (cf. Theorem 3.5) Schematic drawing of G' for $|V| = 6$. Thick edges depict bundles of $m + 1$ parallel 2-paths. **(b)** (cf. Theorem 3.5 and 3.8) The analogously constructed graph for the MPS hardness proof. 3.D

Theorem 3.5. *MPS-BFS is NP-hard.* 3.5

Proof. We give a reduction from Hamiltonian cycle (HC) to MPS-BFS. Let $G = (V, E)$ be an instance for HC, $n := |V|$, $m := |E|$, s a new node, and $B_v := \mathcal{B}^{m+1}_{v,s}$ for each $v \in V$. We construct an input graph G' for MPS-BFS, where

$$\begin{aligned} G' &:= (V', E') & E' &:= E_r \cup E \cup E_\mathrm{B}, \\ V' &:= \{r\} \cup V \cup V_\mathrm{B}, & E_r &:= \{rv \mid v \in V\}, \\ V_\mathrm{B} &:= \bigcup\nolimits_{v \in V} V(B_v), & E_\mathrm{B} &:= \bigcup\nolimits_{v \in V} E(B_v), \end{aligned}$$

cf. Figure 3.D(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': 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.

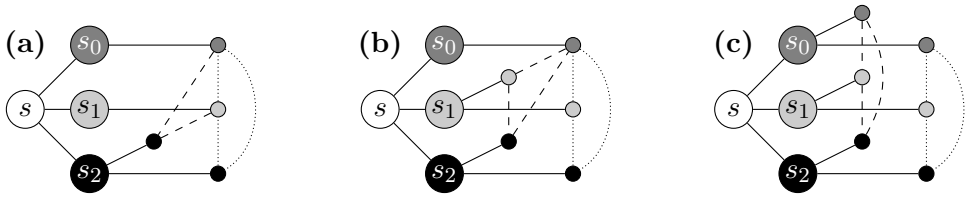
We show that G has a Hamiltonian cycle if and only if G' has a planar subgraph of size $\xi := n(2m + 4)$ that contains T.

(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 $H := G'[T \cup C \cup E_\mathrm{B}]$ 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 $G'[E_\mathrm{B}]$ is planar and contains a face with all nodes of V that allows an arbitrary ordering of those nodes. □

Corollary. *There are (infinitely many) graphs G that allow a BFS tree T_v for each possible start node v such that MPS-BFS on each (G, T_v) is NP-hard.*

Proof. Let G' be the graph constructed in the proof above. We construct G'' by replacing each edge $rv \in E_r$ with $\mathcal{B}^{m+1}_{r,v}$ (cf. Figure 3.D(b)), and replacing each edge of E with a path containing 5 new edges where each of the 4 new nodes is also connected to r by a new edge. Note that we can reach all nodes of V in at most 4 BFS levels, independently 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. □



3.E **Figure 3.E:** (cf. Theorem 3.6) Arising $K_{3,3}$-subdivisions after adding two triangles to the BFS tree. One triangle is dotted, the other is dashed. From left to right: **(a)** both triangles share two nodes, **(b)** both triangles share a single node, **(c)** the triangles are disjoint.

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

3.6 **Theorem 3.6.** *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

$$T := \big\{ss_i \mid i \in [3]\big\} \cup \big\{s_{\phi(v)}v \mid v \in V\big\}.$$

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 Figure 3.E. 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:

3.7 **Corollary 3.7.** *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.

3.4 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 from Vertex Cover to a HC restricted to triangle-free graphs and from that to MPS. We give a direct simple reduction from general HC to MPS.

3.8 **Theorem 3.8.** *MPS is NP-hard.*

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

$$E_{\mathrm{B}} := \bigcup\nolimits_{v \in V} (\mathcal{B}^{m+1}_{r,v} \cup \mathcal{B}^{m+1}_{v,s})$$

(cf. Figure 3.D(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

Algorithm 3.F: Cactus Algorithm 3.F

Input: connected simple graph $G = (V, E)$
edge set $S_1 := \varnothing$
while $\exists$ triangle $T \subseteq E$ whose nodes are in 3 different components of (V, S_1) **do**
  $S_1 := S_1 \cup T$
$S_2 := S_1$
while $\exists$ edge $e \in E$ whose nodes are in different components of (V, S_2) **do**
  $S_2 := S_2 \cup \{e\}$
return S_2

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 in G. The graph $H := G'[C \cup E_B]$ 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 3.9. *MPS remains NP-hard for graphs with any given girth.* 3.9

3.5 Algorithms Inspired by Cactus Structures

The (greedy) *Cactus Algorithm*, see Algorithm 3.F, for MPS was developed by Călinescu et al. [Căl+98]. It 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 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.

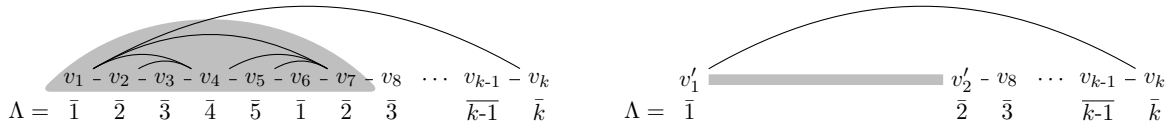
It is fairly natural to ask for a more sophisticated yet easily implementable greedy selection of the triangles to build a cactus. We start with a close relative of the Cactus Algorithm.

Lemma 3.10. *Let $k > 2$ and Γ_k be a cycle on k vertices v_i with $v_1, v_2, \ldots, v_k$ in order. Let $\Lambda(v_i) := i \bmod 5$ be the labels of the vertices. The interior of the cycle can be triangulated with an edge set T such that for all edges $uw \in T$ we have: $\Lambda(u) \neq \Lambda(w)$.* 3.10

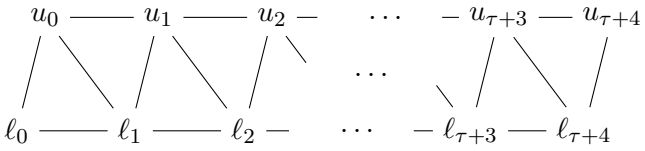
Proof. If $k \leq 5$, the labels of all v_i are different. Thus, any triangulating edge set satisfies the property.

Let now $k > 7$. Among the first seven nodes we define the edge set

$$T' := \{v_1v_4, v_2v_4, v_4v_7, v_5v_7, v_1v_7\}$$



3.G **Figure 3.G:** (cf. Lemma 3.10) Collapsing the edges in T' to a single edge $v'_1v'_2$.



3.H **Figure 3.H:** (cf. Theorem 3.11) The graph R_τ.

where all its edges have endpoints with different labels, c.f. Figure 3.G. Note that for $k = 8$ the set T' is a triangulating edge set with the desired property. If $k > 8$ we collapse all edges and nodes in T' to two new nodes v'_1 and v'_2 and a single edge $v'_1v'_2$, c.f. Figure 3.G. Because $\Lambda(v_1) = 1 = \Lambda(v'_1)$ and $\Lambda(v_7) = 2 = \Lambda(v'_2)$, we can see the graph Γ_k with the collapsed set T' as a smaller instance of the same problem of size $k^{\text{new}} = k^{\text{old}} - 5$. On this smaller instance the problem can now be solved recursively. The solution T is then obtained as the union of the T' sets in each recursion step.

Finally, for $k = 6$ the set $T' \setminus \{v_4v_7, v_1v_7\}$ and for $k = 7$ the set $T' \setminus \{v_1v_7\}$ have the desired property. □

The greedy version of the Cactus Algorithm selects triangles and afterwards connects the thereby arising cactus structures with single edges. We show in the next theorem that a 7/18 bound holds independently of the final selection of single edges.

3.11 **Theorem 3.11.** *Let $\mathcal{A}$ be an algorithm that selects triangles without restrictions. Assume that $\mathcal{A}$ selects a maximal set of triangles. Then, $\mathcal{A}$ has an approximation ratio of at most* 7/18.

Proof. Let τ be divisible by 5 and $R_\tau := (V_\tau, E_\tau)$ be the graph

$$V_\tau := \{u_i, \ell_i : i = 0, 1, \ldots, \tau + 4\}$$
$$E_\tau := \bigcup_{i=0}^{\tau+3} \{u_iu_{i+1}, \ell_i\ell_{i+1}, u_i\ell_{i+1}\} \cup \bigcup_{i=0}^{\tau+4} \{u_i\ell_i\},$$

c.f. Figure 3.H. We add edges $u_2u_{\tau+1}$, $\ell_0\ell_{\tau+4}$ and triangulate the cycles $u_2u_3 \cdots u_{\tau+1}u_2$ and $\ell_0\ell_1 \cdots \ell_{\tau+4}\ell_0$ according to Lemma 3.10 obtaining R'_τ. Consequently, R'_τ contains only a single non-triangular face of degree 9: $u_0u_1u_2u_{\tau+1}u_{\tau+2}u_{\tau+3}u_{\tau+4}\ell_{\tau+4}\ell_0u_0$. Next, we add planarly into each triangular face of R'_τ a new node together with 3 edges to obtain the graph H. This adds $(2|V_\tau| - 4) - (9 - 2) = 4\tau + 9$ nodes, denoted by M. Thus, H is a planar graph that contains $6\tau + 19$ nodes and $(3|V(H)| - 6) - (9 - 3) = 18\tau + 45$ edges.

We extend H to H' by adding

$$\Delta_1 := \big\{u_i\ell_{i-2}, u_i\ell_{i+3}, \ell_{i-2}\ell_{i+3} \mid 2 \le i \le \tau + 1\big\}$$

that resembles 5 node-disjoint chains of triangles each, see Figure 3.I for a visualization. Finally, we construct the input graph G by joining H' with the K_5^-: Let $f_0, \ldots, f_4$ denote 5 distinct faces of K_5^-, and $\delta(f)$ the nodes incident to face f. We add

$$\Delta_2 := \big\{\ell_iv \mid 0 \le i \le 4, v \in \delta(f_i)\big\}.$$

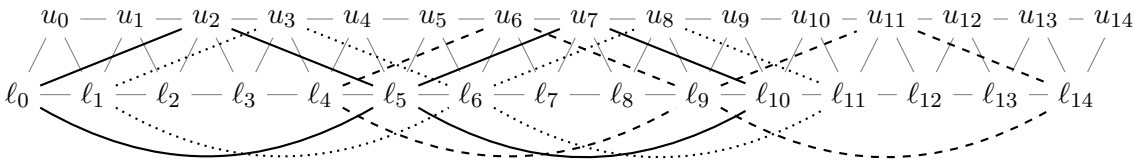


Figure 3.I: (cf. Theorem 3.11) Three (stroked, dotted and dashed) of the five chains of triangles introduced with Δ_1. For simplicity we show them only on R_τ for $\tau = 10$ instead of H. Note that all triangles of a single chain pairwise share a vertex and the chains itself are pair-wise node-disjoint. 3.I

To sum up, we have

$$G = \big(V(H) \cup V(K_5^-),\ E(H) \cup \Delta_1 \cup E(K_5^-) \cup \Delta_2\big)$$

with $21\tau + 69$ edges.

The algorithm may start by picking $S := \Delta_1 \cup \Delta_2 \cup E(K_5^-)$. Note that this results in five chains of $\tau/5$ triangles, each of which can be planarly embedded inside a distinct face of the K_5^-. We observe that $E(G) \setminus S = E(H)$. Any edge in $E(H[V(\Delta_1)])$ connects nodes in different chains by Lemma 3.10. Consequently, adding any of these edges yields a K_5 subdivision and thus non-planarity. A similar observation holds for all 2-paths in the cut $[M, V(\Delta_1)]$: Adding any $V(\Delta_1)$-M-$V(\Delta_1)$-path also yields a non-planar graph. Thus, for any node in M that is adjacent to three nodes of $V(\Delta_1)$ we can add at most a single edge. Note that the number of nodes in M that are adjacent to less than three nodes of $V(\Delta_1)$ is 9. Consequently, any planar subgraph S' containing S has at most

$$\begin{aligned}|E(G)| - 2(|M| - 9) - |E(H[V(\Delta_1)])| &= 21\tau + 69 - 2((4\tau + 9) - 9) - (3(2\tau + 5) - 6 - 5)\\ &= 7\tau + 65\end{aligned}$$

edges. We conclude that the approximation ratio of any algorithm that picks triangles in arbitrary order is at most

$$\lim_{\tau\to\infty} \frac{|E(S')|}{|E(H)|} = \lim_{\tau\to\infty} \frac{7\tau + 65}{18\tau + 45} = \frac{7}{18}.$$

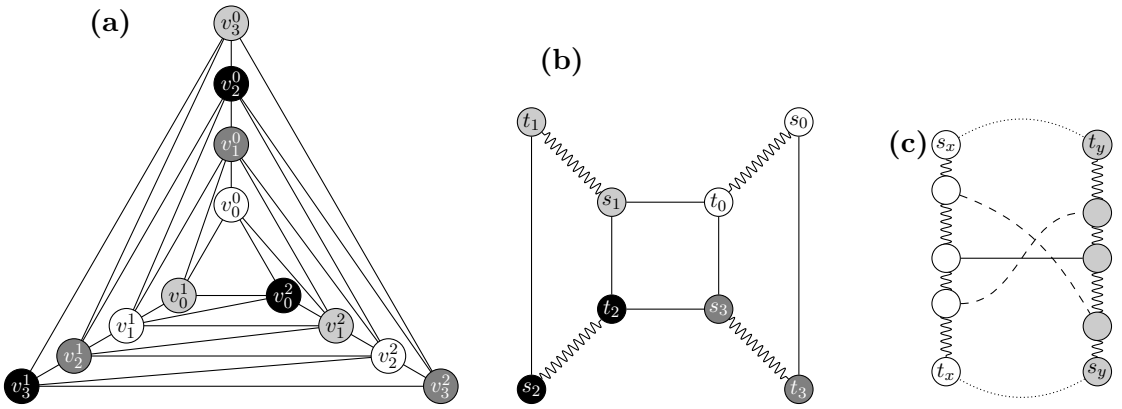
□

Remark. The input graph G in the above proof contains $6\tau + 19$ nodes and $21\tau + 69$ edges. Thus, G has a skewness of at least $3\tau + 18 \in \Theta(|V|)$. The example by [Căl+98] proved the tightness of the 7/18 approximation using a graph family with only constant skewness (a planar graph). Planar worst-case instances can be treated in an additional preprocessing step by using a linear time planarity test. Also the MPS of graphs with constant or bounded skewness μ can be computed in polynomial-time using $\mathcal{O}(|E|^\mu)$—which is constant—planarity tests. ⌟

The following result is another example for a graph family with non-constant skewness that shows the tightness of the 7/18 approximation. Note that it extends the example of [Căl+98] and the arising graphs are very similar.

Lemma 3.12. *Algorithm 3.F has an approximation ratio of at most* 7/18 *even when restricted to graphs $G = (V, E)$ with skewness $\Theta(|V|)$.* 3.12

Proof. We base this proof on the worst-case instance by [Căl+98]. Consider any connected cactus S consisting of p triangles. Note that $|V(S)| = 2p + 1$. Denote with S' a triangulated supergraph of S on $V(S)$. The number of faces in S' equals $4p - 2$. We construct S'' from S' by adding a node into each face of S' along with 3 new edges. Thus, S'' is a planar triangulated graph on



3.J **Figure 3.J:** (cf. Theorem 3.13) **(a)** The graph H_p for $p = 4$. **(b)** The outerplanar graph X_p on $V(H_p)$. **(c)** Inserting independent edges whose endpoints are non-adjacent between V_x and V_y in X_p.

$6p - 1$ nodes. The graph S'' is the example constructed in [Căl+98]. To obtain the input graph G, we add a single node v together with $4p - 2$ new edges such that v is adjacent to all nodes in $V(S'') \setminus V(S)$. Since G contains $6p$ nodes and $22p - 11$ edges, it has a skewness of at least $4p - 5 \in \Theta(|V|)$. Clearly, the number of triangles in G remains equal to the number of triangles in S''. Consequently, the algorithm may pick the cactus S together with a single edge for each remaining node. Thus, the approximation ratio is bounded from above by

$$\frac{|E(S)| + |V \setminus V(S)|}{|E(S'')|} = \frac{7p - 1}{18p - 9}.$$ □

Now, we investigate algorithms that greedily select either edges or triangles in an "intuitively smart" manner.

Definition (forbidden edge (set)). 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).

3.13 **Theorem 3.13.** *GES has a tight approximation ratio of* $1/3$.

Proof. Let $p \geq 4$. Define $H_p := (V, E_H)$ with

$$V := \left\{v_\ell^i \mid \ell \in [p], i \in [3]\right\}$$
$$E_H := \left\{v_\ell^i v_\ell^{i+1} \mid \ell \in [p], i \in [3]\right\} \cup \left\{v_{\ell-1}^i v_\ell^i, v_\ell^i v_{\ell-1}^{i+1} \mid 1 \leq \ell \leq p-1, i \in [3]\right\}$$

cf. Figure 3.J(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 $\phi(v_\ell^i) := (3\ell + i) \bmod 4$. For any color $c \in [4]$, let $V_c := \{v \in V \mid \phi(v) = c\}$ 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 $P_c := \{\pi_i^c \pi_{i+1}^c \mid 1 \leq i < |V_c|\}$. 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

$$X_p := \Big(V,\ \{t_0 s_1, s_1 t_2, t_2 s_3, s_3 t_0, s_0 t_3, t_1 s_2\} \cup \bigcup\nolimits_{c \in [4]} E(P_c)\Big)$$

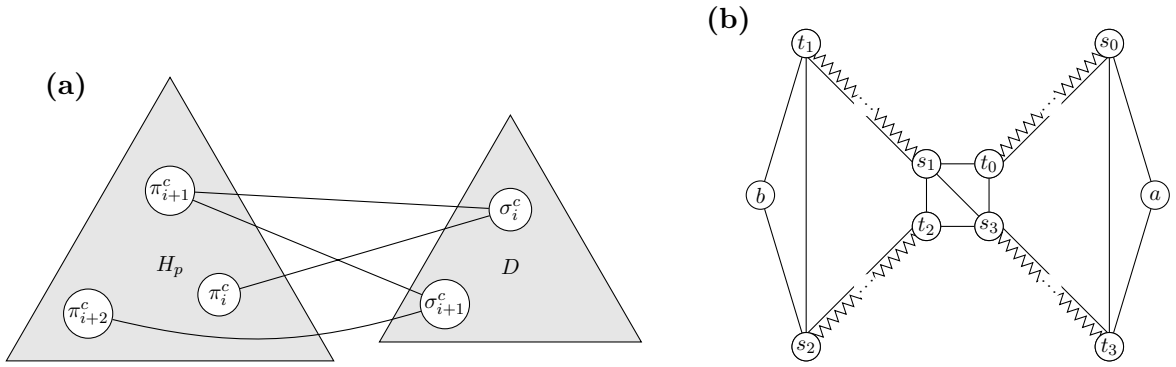


Figure 3.K: (cf. Theorem 3.14) **(a)** Schematic structure of G' showing only some nodes of color c. **(b)** The outerplanar graph X'_p. 3.K

adding six edges to the paths P_c, cf. Figure 3.J(b).

Consider the graph $G := H_p \cup X_p$ (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

$$\lim_{p\to\infty} \frac{|E(X_p)| + \text{const}}{|E(H_p)|} = \frac{1}{3}.$$

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. Figure 3.J(c). The number of color pairs is constant. Thus, for any color pair (x, y) and suitable face, 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 3.14. *Any algorithm that selects the edges picked by GTS has an approximation ratio of at most* 7/18. 3.14

Remark. Note the difference between this result and Theorem 3.11: In Theorem 3.11 we considered algorithms that do not apply any restriction on the choice of the selected triangles. The theorem above is about algorithms that use a greedy strategy with respect to the thereby arising forbidden triangles. ⌟

Proof. Let G be the graph of the proof of Theorem 3.13 for an arbitrary but fixed $p \geq 5$, and $n_p := |V(G)| = 3p$. 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:

3.L **Algorithm 3.L:** Dense Subgraph Selection (DSS)

Input: parameter $k \in \mathbb{N}_{\geq 3}$, connected simple graph $G = (V, E)$
edge set $S := \varnothing$
while S is not maximal planar **do**
  Find a planar subgraph Q with up to k nodes W such that
  (i) $S[W] \subsetneq E(Q)$,
  (ii) $S \cup E(Q)$ is planar,
  (iii) Q has maximum density among all subgraphs that satisfy (i) and (ii), and
  (iv) possibly further restrictions (see text).
  $S := S \cup E(Q)$
return S

Let D be a copy of H_{p-1} where we delete the node v^2_{p-2}. Note that D is triangulated with the exception of one face of degree 5. As in the proof above, this graph is 4-colorable which induces the node partitions $D_c := V_c(D)$ 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

$$X'_p := \big(V(X_p) \cup V(D) \cup \{a, b\},\ E(X_p) \cup E_\triangle \cup \{s_1 s_3, s_0 a, a t_3, t_1 b, b s_2\}\big)$$

with $E_\triangle := \{\pi^c_i \sigma^c_i, \sigma^c_i \pi^c_{i+1} \mid c \in [4], \pi^c_i \pi^c_{i+1} \in P_c\}$, see Figure 3.K. That is, $E_\triangle$ consists of a level-monotone Hamiltonian path for each color class.

Let

$$G' := \big(V(X'_p),\ E(X'_p) \cup E(H_p) \cup E(D)\big).$$

The graph $J := H_p \cup D$ 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 3.13, 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 $L := \{v_f \mid f \in F_J\}$ 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 $|L|$ plus a constant: 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

$$\lim_{p\to\infty} \frac{|E(X'_p)| + |L| + \text{const}}{|E(J)| + |E_L|} = \lim_{p\to\infty} \frac{(3n_p + \text{const}) + (4n_p + \text{const}) + \text{const}}{(6n_p + \text{const}) + (12n_p + \text{const})} = \frac{7}{18}. \qquad \square$$

Similar to any DFS- and BFS-based algorithm, 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 3.16.

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.

We denote Algorithm 3.L 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 3.15. *Consider any MPS instance G. It remains NP-hard to find a maximum planar subgraph of G under the restriction that it contains the solution S of DSS-D.* 3.15

Proof. 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 $G_v := G'[\{v\} \cup V_v]$ is triangulated. Let $S := \bigcup_{v \in V} E(G_v)$. 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 3.9. □

Theorem 3.16. *Consider any MPS instance G. It remains NP-hard to find a maximum planar subgraph of G under the restriction that it contains the solution S of DSS-C.* 3.16

Proof. Consider a graph G together with a spanning tree T. We know from Corollary 3.7 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 Algorithm 3.L 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, independently of k. □

Note that Theorem 3.16 for $k = 3$ is the above claimed hardness result for algorithms based on triangle selection.

Theorem 3.17. *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 $1/2$.* 3.17

Proof. First assume that $k \geq 7$. Let $F := \{f_0, \ldots, f_3\}$ denote the set of faces of a K_4, δ_i the set of nodes incident to face f_i and $\kappa := k - 7$. We define $F' := F \setminus \{f_0\}$ and $\{b, t, u_0\} =: \delta_0$. We construct $G = (V, E)$ with

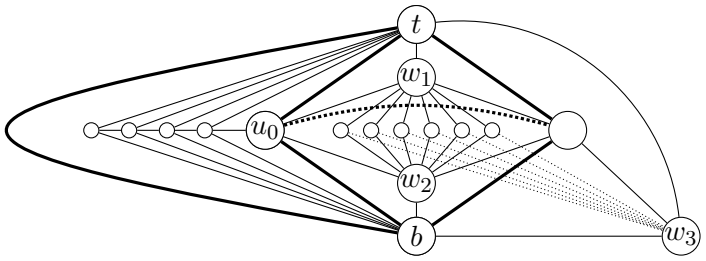
$$\begin{aligned} V &:= V(K_4) \cup \{w_i \mid f_i \in F'\} \cup \{u_{i+1} \mid i \in [\kappa]\}, \\ E &:= E(K_4) \cup E_W \cup E_U, \\ E_W &:= \{w_i v \mid f_i \in F', v \in \delta_i\}, \\ E_U &:= \textstyle\bigcup_{i=1}^{\kappa} \{bu_i, u_i t, u_i u_{i-1}\}. \end{aligned}$$

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 $L := \{s_1, \ldots, s_\ell\}$ and $E_L := \bigcup_{i \in [\ell]} \{s_i w_1, s_i w_2, s_i w_3\}$ (cf. Figure 3.M), 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 therefore 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

$$\lim_{\ell \to \infty} \frac{|Q| + \ell}{|H'|} = \lim_{\ell \to \infty} \frac{|E| + \ell}{|E| - 1 + 2\ell} = \frac{1}{2}.$$

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



3.M **Figure 3.M:** (cf. Theorem 3.17) Schematic drawing of the input graph for $\ell = 6$ and $k = 4$. The K_4 subgraph is highlighted by thick edges. Dotted edges are not included in H_2.

3.6 Algorithms Based on Decomposition

This section collects some notes on algorithmic ideas based on graph decompositions. It is known [ALS91, Theorem 3.5] that MPS can be expressed in extended monadic second order logic. This implies the existence of a linear-time algorithm [ALS91, Theorem 5.6] for solving MPS on graphs with bounded tree-width. In the text below we present some insights regarding the cut- and path-width.

Note that the focus in this section is on the usage of decompositions that answer questions naturally arising when constructing approximation algorithms, such as *How can we bound the number of edges between these sets of nodes.* The presented results show examples of such usecases.

Definition (numbering, cut-width). Given a graph $G = (V, E)$, $|V| = n$, a *numbering* of G is a bijective mapping $\pi\colon V \to [n]$. The *cut-width of* π is

$$\max_{p\in[n-1]} \big|\{uv \in E \mid \pi(u) \le p < \pi(v)\}\big|.$$

The *cut-width* $\mathrm{cw}(G)$ *of* G is the minimum cut-width over all numberings. ⌟

See [KS93] for details. Note that the cut-width is bounded from below by the path-width [Bod86, Theorem 3.10] that is bounded from below by the tree-width since each path decomposition is a tree decomposition.

3.18 **Lemma 3.18.** *Let G be a connected graph and π be a numbering with cut-width $c := \mathrm{cw}(G)$. For each $\epsilon > 0$, there is a $(1-\epsilon)$-approximation algorithm for MPS with run-time $\mathcal{O}(|V| \cdot f(c/\epsilon))$, where f is a computable function independent of the instance G.*

Proof. Let H be a maximum planar subgraph of G, $\hat{m} := |E(H)|$, $b := \lceil |V|\epsilon/c \rceil$ and

$$B_i := \{v \mid ic/\epsilon \le \pi(v) < (i+1)c/\epsilon\}$$

be a partition of the ordered vertices into b bags. W.l.o.g. $\hat{m} \ge |V|$. For each B_i, we compute a MPS (B_i, E_i) in $\mathcal{O}(f(c/\epsilon))$ time. Let M be the union of $E_1, \ldots, E_b$. Any cut along π contains at most c edges. Thus, the union of all cuts between adjacent B_i's contains at most $c(b-1) \le |V|\epsilon$ edges. Since we find optimal solutions on all B_i, we obtain a planar subgraph of G that contains at least $\hat{m} - |V|\epsilon$ edges. We obtain a $(1-\epsilon)$-approximation using M since

$$\frac{|M|}{\hat{m}} \ge \frac{\hat{m} - |V|\epsilon}{\hat{m}} = 1 - \frac{|V|\epsilon}{\hat{m}} \ge 1 - \epsilon. \qquad \square$$

We now focus on the path-width and try to find characteristica of maximal planar subgraphs in graphs with bounded path-width.

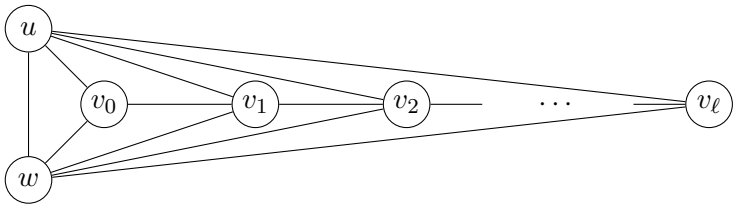


Figure 3.N: The planar triangulated graph $G = (\{u, w, v_0, v_1, \ldots, v_\ell\}, E)$ with path-width three (bags $B_i := \{u, w, v_{i-1}, v_i\}$). 3.N

Definition (path-width, path decomposition). [Kin92; RS83] Given a graph $G = (V, E)$, $|V| = n$, and a numbering π of G. Define

$$V_\pi(i) := \{u \in V : \pi(u) \leq i \text{ and there is some } v \in V \text{ such that } uv \in E \text{ and } \pi(v) > i\}.$$

The *vertex separation number of* π is $\max_{p\in[n-1]} |V_\pi(i)|$. The *path-width* $\mathrm{pw}(G)$ *of* G is the minimum vertex separation number over all the numberings. A *path decomposition* is a tree decomposition $(\mathcal{B}, T)$ where the underlying tree T is a path. ⌟

Graphs with path-width $k \leq 2$ have at most $2n - 3$ edges, which follows directly from the definition. Thus, they can be 1/2-approximated using spanning trees.

However, there are planar triangulated graphs with path-width $k = 3$. Consider the graph $G := (V, E)$ shown in Figure 3.N on $3 + \ell$ nodes with $V := \{u, w, v_0, \ldots, v_\ell\}$, and

$$E := \{uw,\ v_0u,\ v_0w\} \cup \{v_iv_{i-1},\ v_iu,\ v_iw \mid 1 \leq i \leq \ell\}.$$

It is easy to see that G is triangulated and that a path decomposition of G of width 3 is given by $B_i := \{u, w, v_{i-1}, v_i\}$ for $1 \leq i \leq \ell$.

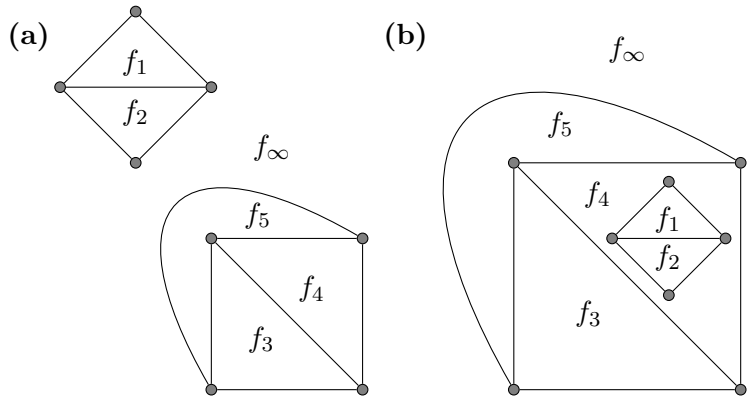
3.7 Alternative Proof for the Cactus Algorithm

In this section we present an alternative proof for the result that the Cactus Algorithm has an approximation ratio of 7/18. We developed this proof because we were unable to apply the existing one by Călinescu et al. to new classes of algorithms. Our goal was to use the new method to prove that other greedy approximation algorithms achieve a better approximation ratio. Unfortunately, we did not find a better algorithm. Nevertheless, we hope that the underlying ideas are useful in the future to build proofs for new approximation algorithms on them.

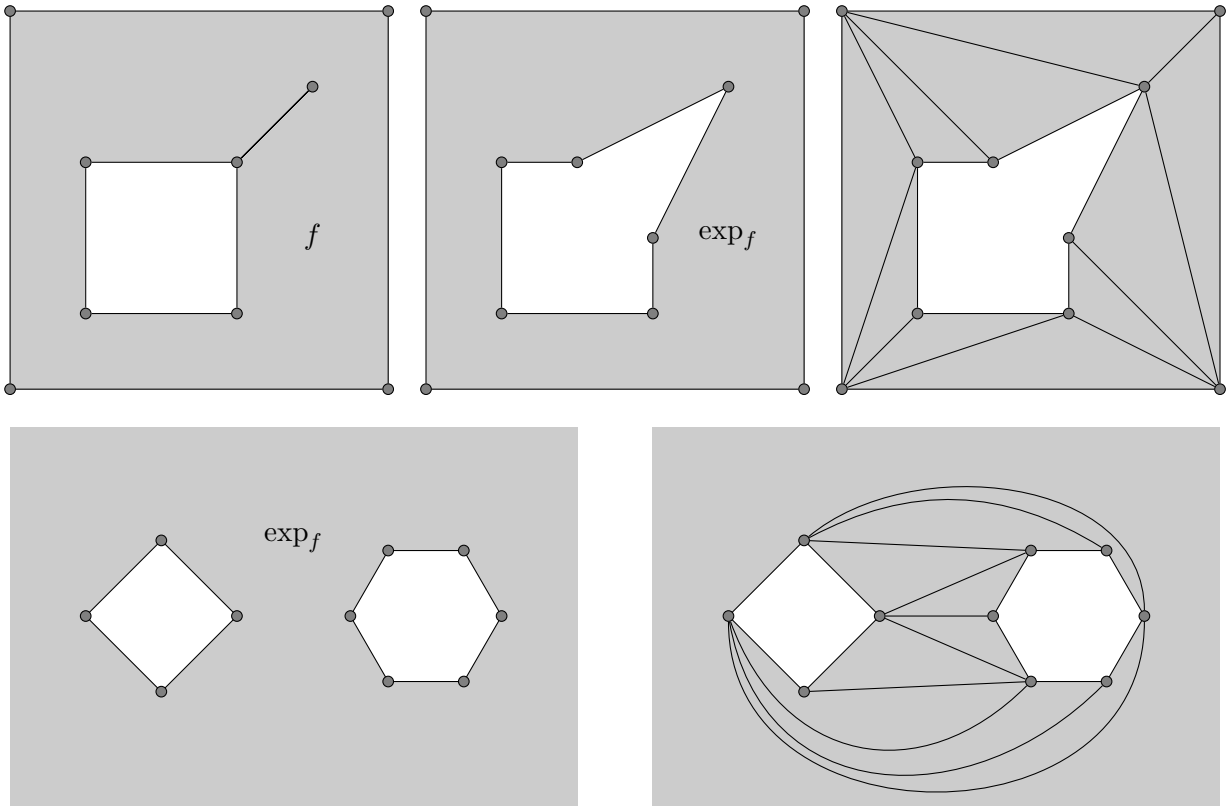
Definition (extended topological embedding). [Cha$^+$15; JS09] Let G be a planar graph. We define a *extended topological embedding* as follows: We specify a planar embedding for each connected component which determines sets of non-outer faces. For each connected component of G we specify a containing face, which may be a non-outer face of some other component or the unique outer face f_∞. Cycles of containment are forbidden.[1] A face $f = \{w_1, \ldots, w_p\}$ in a topological embedding has several facial walks w_i along its boundary. The size $|w|$ of a facial walk w is defined as the number of nodes of w, where we count node repetitions. The size $|f|$ of a face is defined as $|f| := \sum_{w\in f} |w|$. ⌟

Remark. The number of walks in a face equals the number of connected components incident to the face. ⌟

Example. Consider the graph $K_4 \cup K_4^-$. The figures 3.O(a) and 3.O(b) show two topological embeddings of U with faces that consist of more than a single facial walk. ⌟



3.O **Figure 3.O: (a)** A topological embedding of the union $U := K_4 \cup K_4^-$ where the containing face of both components is the outer face f_∞. Note that f_∞ consists of two facial walks. **(b)** A topological embedding of U where the containing face of the K_4^--component is the non-outer face f_4 of the K_4-component. Here, f_∞ is only a single facial walk but f_4 consists of two facial walks.



3.P **Figure 3.P:** Top left: A face f in a topological embedding of a graph. Top middle: The expansion $\exp_f$ of f drawn similar to the initial embedding. Top right: A holey triangulation H_f of f. Bottom left: The expansion $\exp_f$ of f drawn as described in the definition. Bottom right: The same holey triangulation H_f drawn consistent with the definition.

Definition (expansion). A given face $f = \{w_1, \ldots, w_p\}$ can be identified with the outer region of p cycles with lengths $|w_1|, \ldots, |w_p|$, respectively, embedded side by side in the plane. We call this region the *expansion* $\exp_f$ of f. ⌟

Definition (holey triangulation). Let f be a given face. A triangulation of $\exp_f$ is called *holey triangulation (HT)* of f. ⌟

Example. Consider the graph in Figure 3.P with the given topological embedding. The figure visualizes the expansion $\exp_f$ of the face f and a possible holey triangulation. ⌟

Lemma 3.19. *A HT of a face $f = \{w_1, \ldots, w_p\}$ adds $\tau(f) := |f| + 3p - 6$ edges.* 3.19

Proof. A triangulated graph on $|f|$ nodes has $3|f| - 6$ edges. Discounting the $\sum_{w \in f}(|w| - 3) = |f| - 3p$ edges in the interior of the cycles, we obtain

$$(3|f| - 6) - |f| - (|f| - 3p) = |f| + 3p - 6$$

new edges. □

Theorem 3.20. *The approximation ratio of Algorithm 3.F is at least* 7/18. 3.20

Proof. Sketch of the proof: Let $H = (V, E_H)$ denote a maximum planar subgraph of G. The algorithmic solution S_1 allows us to consider two classes of faces of a certain subgraph $H[C]$ of H. To reconstruct H from $H[C]$, we insert the remaining nodes—with all incident edges in H—while bounding the maximum number of reconstructed edges. This allows us to bound the total number of edges in H from above which we will relate to the number of edges in S_2.

Let $C_1, \ldots, C_k \subseteq V$ denote the nodes of the connected components of $G[S_1]$, $C := \bigcup_{i=1}^{k} C_i$, and $L := V \setminus C$. We say that a face f of $H[C]$ is *free* if and only if for each cycle $w \in f$ there is a component K of $G[S_1]$ such that all nodes of w are in K.

Claim 3.21. *Given a face $f = \{w_1, \ldots, w_p\}$ in $H[C]$, we reconstruct all ℓ_f nodes from L that are within f in H. The number r_f of reconstructed edges is bounded from above via the inequality* 3.21

$$r_f \leq 2\ell_f + \tau(f) + \mathbb{1}_{\text{free}}(f).$$

Here, $\mathbb{1}_{\text{free}}(f) = 1$ if f is free and 0 otherwise.

We will prove this claim later. Now we use the claim to continue with our main proof.

We define $\delta \geq 0$ such that $|E(H[C])| = 3|C| - 6 - \delta$. We denote the number of free faces that contain nodes of L by ϕ. Note that $|L| \geq \phi$. We can bound the number r of reconstructed edges when reconstructing all nodes from L by

$$r \;=\; \sum\nolimits_{f:\ \ell_f > 0} r_f \;\leq\; \phi + \sum\nolimits_{f:\ \ell_f > 0} 2\ell_f + \tau(f) \;\leq\; \phi + 2|L| + \sum\nolimits_{f} \tau(f) \;\leq\; \phi + 2|L| + \delta$$

where the last inequality follows from Lemma 3.19: $\sum_f \tau(f)$ is bounded by δ, the number of missing edges for a triangulation of $H[C]$.

We can bound the total number of edges in H by

$$|E_H| \leq (3|C| - 6 - \delta) + r \leq 3|C| + 2|L| + \phi - 6.$$

[1] If the containing face of a component C_1 is a non-outer face of a component C_2 whose containing face is non-outer face of C_1, we get a cycle.

Note that the number of edges in S_2 is $\frac{3}{2}|C| - k/2 + |L| - 1$, since S_2 resembles a spanning tree together with the triangles in the components C_i of S_1 and each component contains $\frac{3}{2}|C_i| - \frac{3}{2}$ edges in the algorithmic solution. It is now easy to see that $|S_2|/|E_H| \geq 7/18$:

Assuming $|S_2|/|E_H| < 7/18$ leads to $6|C| - 9k + 4|L| + 24 < 7\phi$. Clearly, the number of free faces is at most

$$2|C| - 4k = \sum_{i=1}^{k} (2|C_i| - 4)$$

if all $H[C_i]$ would be triangulated. Therefore, $\phi \leq 2|C| - 4k$. It follows that $3\phi + 3k + 4|L| + 24 < 7\phi$, which results in $4(|L| - \phi) < -3k - 24$. Thus, $|L| < \phi$, a contradiction. □

Proof (of Claim 3.21). The number of edges inserted in $\exp_f$ of f while reconstructing the ℓ_f nodes is an upper bound on the number of reconstructed edges in f itself. We say that a triangle is *forbidden* if and only if all of its nodes lie in different components of $H[C]$. The algorithm would have found such triangles. Consequently, when inserting the ℓ_f nodes, no forbidden triangles can be created. We will now consider $\exp_f$ that we insert ℓ_f nodes into. Assuming a triangulation Δ of this region, we bound the number of forbidden triangles in Δ from below. This yields an upper bound on the number of reconstructed edges in f.

A triangulated graph on $|f| + \ell_f$ nodes has $3(|f| + \ell_f) - 6$ edges and $2(|f| + \ell_f) - 4$ triangles. The expansion $\exp_f$ itself contains $|f|$ edges. The interior triangulations of the cycles of lengths $|w_1|, \ldots, |w_p|$ have overall

$$\sum_{i=1}^{p} (|w_i| - 3) = |f| - 3p$$

edges. There remain

$$3(|f| + \ell_f) - 6 - |f| - (|f| - 3p) = |f| + 3\ell_f - 6 + 3p = 3\ell_f + \tau(f)$$

edges in Δ. These interior triangulations together with $\exp_f$ induce

$$\sum_{i=1}^{p} (|w_i| - 2) = |f| - 2p$$

triangles. With each non-forbidden triangle in Δ we can associate a unique edge of $\exp_f$ where both incident nodes lie in the same component of $H[C]$. Thus, there are at most $|f|$ allowed triangles in a free face. A non-free face has at least two edges whose incident nodes are in different components. Such an edge cannot be part of a triangle with a node of L because this would be a forbidden triangle. Consequently, at least

$$2(|f| + \ell_f) - 4 - (|f| - 2p) - (|f| - 2 + 2 \cdot \mathbb{1}_{\text{free}}(f)) = 2\ell_f - 2 + 2p - 2 \cdot \mathbb{1}_{\text{free}}(f)$$

triangles are forbidden. Note that it is possible to choose Δ such that all edges of H within f are in Δ. Any edge in Δ that we do not reconstruct, destroys at most 2 forbidden triangles.

Combining the results above we conclude

$$r_f \leq (3\ell_f + \tau(f)) - \tfrac{1}{2}(2\ell_f - 2 + 2p - 2 \cdot \mathbb{1}_{\text{free}}(f)) \leq 2\ell_f + \tau(f) + \mathbb{1}_{\text{free}}(f). \qquad \square$$

3.8 Summary and Conclusion

Since 1998 the best known approximation algorithm [Căl+98] for MPS is an almost non-implementable enhanced version of a greedy algorithm searching for cactus structures. Our initial goal was to develop an approximation algorithm that breaks the even today holding bound of 4/9. During our non-successful attempts we gained more and more knowledge about limits of existing algorithms and possible generalizations or new variants.

We established alternative—and in most cases shorter—proofs for the approximation ratio by maximal planar subgraphs, the hardness of MPS itself and for the 7/18 bound of the greedy version of the Cactus Algorithm.

Based on state-of-the-art planarity testing algorithm we introduced the problems MPS-DFS and MPS-BFS. We were able to show the hardness of the corresponding new problems, too. Furthermore, a 2/3 bound for approximation ratio for these class of algorithms was carved out.

Moreover, we showed the independence of the final single edge selection step of any algorithm based on triangle selection without restrictions. The transition from triangle selection to the selection of larger dense subgraphs results in a bound of 1/2 for the approximation ratio of greedy dense subgraph selection algorithms.

We were not able to break the 4/9 barrier with a new approximation algorithm. This remains as an open task for future work. Even the new algorithm by Chalermsook and Schmid [CS17] does not achieve 4/9 as a lower bound. But note that the methods used to show the approximation ratio of 13/33 of their Diamond Selection Algorithm seem to contain gaps that can be used to prove even better lower bounds. So far, no worst-case examples for the Diamond selection are known.

Chapter 4

Exact Algorithms for the Maximum Planar Subgraph Problem

In this chapter we consider exact algorithms for the Maximum Planar Subgraph problem. The aim of our work is to study formulations based on different planarity criteria. Examples for other approaches are specialized branch-and-cut techniques as discussed in Jünger and Mutzel [JM96].

As highlighted in Chapter 1 there are various applications for the MPS problem in graph drawing, facility layout, and the layout of electronic circuits.

We give a non-complete list of 14 planarity criteria where we picked four to build exact algorithms on them: *Additional Minors* for *not apex* graphs, *Total Orders* and the Dushnik-Miller dimension, *Theta Graphs*, and *Facial Walks*.

In most cases, we only discuss ideas either for ILP or PBS formulations. But most of the ideas can be expressed in ILP *and* PBS formulation. For the sake of brevity, we do not give all formulations explicitly.

We evaluate the overall practicality of our formulations in experiments on the *Rome*, *North*, *SteinLib* and *Expander* instances. None of the presented approaches are favorable compared to the standard method using Kuratowski subdivisions.

4.1 A Summary of Known Planarity Criteria

Besides the famous K_5-$K_{3,3}$-subdivision and K_5-$K_{3,3}$-minor criteria by Kuratowski [Kur30] (see Section 4.2 (p. 69) for details) and Wagner [Wag37] there are many more criteria that are more or less useful for ILP/SAT-based exact MPS algorithms. We give a (non-complete) list[1] of criteria that we used to build exact algorithms or explain why we refrained from using the characterization.

1. Archdeacon and Širáň [AŠ98]: *Characterizing Planarity Using Theta Graphs*:

   Original Abstract: A theta graph is a homeomorph of $K_{2,3}$. In an embedded planar graph the local rotation at one degree-three vertex of a theta graph determines the local rotation at the other degree-three vertex. Using this observation, we give a characterization of planar graphs in terms of balance in an associated signed graph whose vertices are $K_{1,3}$ subgraphs and whose edges correspond to theta graphs.

   See Section 4.5 (p. 78) for details.

2. Došen and Petrić [DP15]: *A planarity criterion for graphs*:

[1] The bibliography of Little and Sanjith [LS10] *Another characterisation of planar graphs* and the introduction of Thomassen [Tho80] *Planarity and Duality of Finite and Infinite Graphs* are the primary sources of this list.

MathSciNet Review by R. Bruce Richter: Kuratowski famously characterized planar graphs in 1930 as having no subdivision of either $K_{3,3}$ or K_5. Since then, there have been many other characterizations of planar graphs. This article presents a characterization in terms of the cyclic ordering of each of the cocycles (these are the minimal edge-cuts and are now usually called "bonds"). The cyclic ordering is, of course, determined by a planar embedding. Graph-theoretically, suppose e_1, e_2, e_3, e_4 is a cyclic ordering of 4 of the edges in a (possibly larger) cocycle. This ordering is disallowed if, on at least one side of the cocycle, there are disjoint paths, one joining the ends of e_1 and e_3, the other joining the ends of e_2 and e_4. The graph G is planar if and only if every cocycle has a cyclic order with no disallowed cyclic subsequence of length 4. The proof is a straightforward application of Kuratowski's Theorem.

The presented criterion is more complex than the criterion by [AŠ98]. We therefore refrain from using it for an exact algorithm.

3. Lipton et al. [Lip+16]: *Six variations on a theme: almost planar graphs*:

   Original Abstract: A graph is apex if it can be made planar by deleting a vertex, that is, $\exists v$ such that $G - v$ is planar. We define the related notions of edge apex, $\exists e$ such that $G - e$ is planar, and contraction apex, $\exists e$ such that G/e is planar, as well as the analogues with a universal quantifier: $\forall v$, $G - v$ planar; $\forall e$, $G - e$ planar; and $\forall e$, G/e planar. The Graph Minor Theorem of Robertson and Seymour ensures that each of these six gives rise to a finite set of obstruction graphs. For the three definitions with universal quantifiers we determine this set. For the remaining properties, apex, edge apex, and contraction apex, we show there are at least 36, 55, and 82 obstruction graphs respectively. We give two similar approaches to almost nonplanar ($\exists e$, $G + e$ is nonplanar and $\forall e$, $G + e$ is nonplanar) and determine the corresponding minor minimal graphs.

   See Section 4.3 (p. 73) for details.

4. Colin de Verdière [Col93]: *On a new graph invariant and a criterion for planarity*:

   MathSciNet Review by Arthur T. White: Let G be a finite connected graph of order v. Let M_G denote the set of symmetric real $v \times v$ matrices $A = (a_{ij})$ satisfying (i) $a_{ij} < 0$ if $\{i, j\} \in E(G)$; and (ii) $a_{ij} = 0$ if $i \neq j$ and $\{i, j\} \notin E(G)$. Then $\mu(G)$ is defined to be the greatest integer n_0 for which there exists $A_0 \in M_G$ for which the second eigenvalue is of multiplicity n_0 and satisfies the strong Arnol'd hypothesis [V. I. Arnol'd, Funktsional. Anal. i Prilozhen. 6 (1972), no. 2, 12–20]. The monotonicity of μ with respect to operations of deletion and contraction is studied. Among the theorems proved are the following: (1) G is planar if and only if $\mu(G) \leq 3$; (2) G is outerplanar if and only if $\mu(G) \leq 2$. It is conjectured that always $\mu(G) \geq \chi(G) - 1$; this, in conjunction with (1), would immediately give the four-color theorem.

   The algebraic nature of the presented property prevents us from considering this as a base for an exact algorithm.

5. Jaeger [Jae79]: *Interval matroids and graphs*:

   Original Abstract: A base of the cycle space of a binary matroid M on E is said to be convex if its elements can be totally ordered in such a way that for every $e \in E$ the set of elements of the base containing e is an interval. We show that a binary matroid is cographic iff it has a convex base of cycles; equivalently, graphic matroids can be represented as "interval matroids" (matroids associated in a natural way to interval systems). As a consequence, we obtain characterizations of planar graphs and cubic cyclically-4-edge-connected planar graphs in terms of convex bases of cycles.

   The presented complex criterion refrains us from using it for an exact algorithm.

6. Tutte [Tut63]: *How to draw a graph*:

   Summary by Thomassen [Tho80]: "a 3-connected graph is planar if and only if every edge is contained in precisely two induced non-separating cycles"

   The MPS of a given graphs does not need to be 3-connected, thus we cannot use this criterion.

7. Schnyder [Sch89]: *Planar Graphs and Poset Dimension*:

   Original Abstract: We view the incidence relation of a graph $G = (V, E)$ as an order relation on its vertices and edges, i.e. $a <_G b$ if and only if a is a vertex and b is an edge incident on a. This leads to the definition of the order-dimension of G as the minimum number of total orders on $V \cup E$ whose intersection is $<_G$. Our main result is the characterization of planar graphs as the graphs whose order-dimension does not exceed three. Strong versions of several known properties of planar graphs are implied by this characterization. These properties include: each planar graph has arboricity at most three and each planar graph has a plane embedding whose edges are straight line segments. A nice feature of this embedding is that the coordinates of the vertices have a purely combinatorial meaning.

   See Section 4.4 (p. 74) for details.

8. Fournier [Fou74]: *Une Relation de Séparation entre Cocircuits d'un Matroide*:

   MathSciNet Review by W. Dorfler: Let M be a matroid on the set E, and let T, A be subsets of E. Then T is said to separate A in M if A meets at least two components of $M \times (E - T)$. Here $M \times B$ has as circuits those circuits of M that are contained in $B \subset E$. For the bond-matroid M of a graph G, the fact that a cocircuit does not separate E or the union of two cocircuits can be translated into graph-theoretic language. Main result: A matroid is graphical if and only if at least one of any three cocircuits with a nonvoid intersection separates the union of the other two cocircuits. The proof of sufficiency depends on Tutte's characterization of binary and graphical matroids by excluded minors. This result applies to graphs and yields an interesting planarity criterion for simple graphs. Similar characterizations hold for cographic, planar and trivial matroids. Theorem: For a binary matroid to be unimodular it is sufficient that at least one of any 4 circuits or 4 cocircuits with a nonvoid intersection separate the union of the remaining ones. Theorem: A matroid is binary if and only if given 3 circuits C_1, C_2, C_3 with nonvoid intersection for at least one of them, say C_1, then $C_2 - C_1 \neq C_3 - C_1$.

   The criterion itself is given in Corollaire 2.2: *Un graphe simple est planaire si et seulement si quels que soient 3 cycles élémentaires ayant une arête commune et tels qu'aucune arête de l'un n'est une corde d'un autre, l'un au moins de ces 3 cycles sépare les deux autres.*

   A translation is [Tho80, Theorem 4.8]: *A simple graph is non-planar if and only if it contains three cycles such that*

   - *they all have an edge in common and*
   - *no edge of one of the cycles is a chord in one of the other cycles, and*
   - *whenever we contract the edge set of one of the cycles, then those edges of the other two cycles which are not contracted belong to the same block of the resulting graph.*

   Due to the geometric nature of the criterion we refrain from using it for an exact algorithm.

9. Tutte [Tut59]: *Matroids and graphs*:

   MathSciNet Review by J. Isbell: "In the original paper on matroids, Hassler Whitney pointed out [Amer. J. Math. 57 (1935), 507–533] that the circuits of any finite graph G define a matroid. We call this the circuit-matroid and its dual the bond-matroid of G. In the present paper we determine a necessary and sufficient condition, in terms of matroid structure, for a given matroid M to be graphic (cographic), that is the bond-matroid (circuit-matroid) of some finite graph. The condition is that M shall be regular and shall not contain, in a sense to be explained, the circuit-matroid (bond-matroid) of a Kuratowski graph." (Author's introduction.) The paper begins with the definition and basic properties of dual matroids. Using much of the theory previously developed by the author [see the article reviewed above] he goes on to define minors of matroids. The minors of a graphic matroid apparently correspond one-one with the subgraphs; at any rate, "contain" in the main theorem means "contain as a minor". The author proves (after the main theorem) that if M_0 is a connected matroid of subsets of a set A and no two points of A lie in exactly the same members of M_0, then a matroid M contains M_0 as a minor if and only if the configuration

associated with M_0 can be embedded by a dimension-preserving semi-lattice isomorphism in the configuration associated with M.

Definition (H-component, overlap graph). Let H be a subgraph of a graph G. An *H-component* of G is either

- an edge in $E(G) \setminus E(H)$ together with its ends joining two vertices of H or
- a connected component of $G \setminus V(H)$ together with all edges and their ends joining this component to H.

The *overlap graph* of H in G is defined as the graph whose vertices are the H-components of G such that two vertices are adjacent if and only if the corresponding H-components overlap. ⌟

Note that this definition of an overlap graph is not the same as in Section 3.3.

Summary by Thomassen [Tho80, Theorem 4.2(b)]: A graph is planar if and only if it contains a cycle whose overlap graph is non-bipartite.

Due to the complexity of the criterion we refrain from using it for an exact algorithm.

In the beginning of our work on exact algorithms for the MPS problem we also considered the following criteria. Due to time constraints we did not examine them in detail.

10. Keir and Richter [KR96]: *Walks through every edge exactly twice. II*:

    Original Abstract: In this paper we develop a theory of sets of walks traversing every edge twice. Archdeacon, Bonnington, and Little proved that a graph G is planar if and only if there is a set of closed walks $\mathcal{W}$ in G travsersing every edge exactly twice such that several sets of edges derived from $\mathcal{W}$ are all cocycles. One consequence of the current work is a simple proof of the ABL theorem.

    Note that this criterion is a generalization of [ABL95] by Archdeacon et al.

11. Williamson [Wil93]: *Canonical forms for cycles in bridge graphs*:

    Original Abstract: Let $G = (V, E)$ be a biconnected graph and let C be a cycle in G. The subgraphs of G identified with the biconnected components of the contraction of C in G are called the bridges of C. Associated with the set of bridges of a cycle C is an auxiliary graphical structure G_C called a bridge graph or an overlap graph. Such auxiliary graphs have provided important insights in classical graph theory, algorithmic graph theory, and complexity theory. In this paper, we use techniques from algorithmic combinatorics and complexity theory to derive canonical forms for cycles in bridge graphs. These canonical forms clarify the relationship between cycles in bridge graphs, the structure of the underlying graph G, and lexicographic order relations on the vertices of attachment of bridges of a cycle. The first canonical form deals with the structure of induced bridge graph cycles of length greater than three. Cycles of length three in bridge graphs are studied from a different point of view, namely that of the characterization of minimal elements in certain related posets: ordered bridge three-cycles (10 minimal elements), bridge three-cycles (5 minimal elements), bridge deletion three-cycles (infinite number, 7 classes), minor order ($K_5, K_{3,3}$), chordal bridge three-cycles (13 minimal elements), contraction poset (5 minimal elements), cycle-minor poset (infinite number, 14 classes). These results, each giving a different insight into the structure of bridge three-cycles, follow as corollaries from the characterization of the 10 minimal elements of the ordered bridge three-cycle poset. This characterization is constructive and may be regarded as an extension of the classical Kuratowski theorem, which follows as a corollary. Algorithms are described for constructing these various minimal elements in time $O(|E|)$ or $O(|V|)$, depending on the case. The first canonical form gives a constructive proof of the result that a graph is nonplanar if and only if it has a cycle C whose bridge graph G_C (alternatively, skew bridge graph) has a three-cycle. An algorithm is described that constructs this three-cycle in time $O(|E|)$. This is best possible.

12. Fraysseix and Rosenstiehl [FR85]: *A characterization of planar graphs by Trémaux orders*:

    MathSciNet Review by Torrence D. Parsons: This paper gives a new characterization of planar graphs. Let G be a finite connected graph without loops. With any depth-first-search spanning tree T, rooted at a vertex r, is associated a (downward towards the root) semilattice order, called the Tremaux order. Binary relations "T-alike" and "T-opposite" are defined on the set of all cotree edges of T, solely by conditions involving the Tremaux order relative to (T, r). It is proved that G is planar if the set of all cotree edges can be partitioned into two classes so that two edges which are T-alike belong to the same class and two edges which are T-opposite are in different classes. The proof starts from an embedding of G in the plane such that every pair of edges which cross is a pair of cotree edges (relative to T) having exactly one crossing point. If G is nonplanar, then there must exist a special type of "crossing pair" of cotree edges, related to the Tremaux order and having no common endvertices. Assuming that the cotree edges may be partitioned into two classes respecting the T-alike and T-opposite relations, it is shown that the set of all such special crossing pairs may be partitioned into "switching sets", which allow the embedding to be modified to a true plane embedding of G (with no crossing of edges). The proof is based on earlier work of W. T. Tutte, R. B. Levow, Y. Liu, and Rosenstiehl [see Tutte, J. Combin. Theory 8 (1970), 45–53; Rosenstiehl, Ann. Discrete Math. 9 (1980), 67–78].

    See also Fraysseix and Rosenstiehl [FR82]: A depth-first-search characterization of planarity.

13. Little and Sanjith [LS10]: *Another characterisation of planar graphs*:

    Original Abstract: A new characterisation of planar graphs is presented. It concerns the structure of the cocycle space of a graph, and is motivated by consideration of the dual of an elementary property enjoyed by sets of circuits in any graph.

14. In 1976 Little [Lit77] conjectured a criterion that was independently proved by Little and Holton [LH85] *No graph has a maximal* 3*-ring of bonds* (based on [LH82]) and Černjak [Čer80] *On Little's conjecture on planar graphs*:

    MathSciNet Review of [LH82] by J. Sedláček: A bond is a minimal nonempty edge cut. Let R be a set of bonds in a graph G. If the edges of G can be oriented so that every bond of R is directed, then R is said to be consistently orientable. A cyclic sequence of bonds $R = (C_0, C_1, \cdots, C_{n-1}), n \geq 3$, is called a ring if (i) R is consistently orientable, (ii) $C_i \cap C_j \neq \varnothing$ if and only if $i = j, i \equiv j + 1 \pmod n$ or $i \equiv j - 1 \pmod n$, and (iii) no edge belongs to more than two bonds of R. The ring R is said to be strict if there do not exist distinct bonds A, B, C with $B \in R$, $C \in R$, $B \cap C = \varnothing$, $A \subseteq B \cup C$. The ring R is maximal if there exists no ring $R' = (C'_0, C'_1, \cdots, C'_{m-1})$ with $\bigcup_{k=0}^{m-1} C'_k \subseteq \bigcup_{i=0}^{n-1} C_i$ and $m > n$. The aim of this paper is to show that every strict maximal ring has an even cardinality.

    Theorem. [LH82, Theorem 1] *A graph is planar if and only if it has no maximal strict odd ring of circuits.*

    Theorem. [Čer80] *A graph is non-planar if and only if it contains a maximal exact ring with an odd numbers of cycles.*

4.2 MPS via Kuratowski Subdivisions

The most commonly used ILP formulation for the MPS problem is based on Kuratowski subdivisions, see Kuratowski [Kur30].

Assume we have a simple, undirected, weighted graph $G = (V, E, w)$ with weights $w\colon E \to \mathbb{R}$. Let $\mathcal{K}_G$ be the set of all Kuratowski subdivisions of G. We obtain an MPS by deleting an edge from each Kuratowski subdivision. We use binary variables x_e to denote if edge e is deleted from G in the resulting MPS. The following formulations results in an MPS of maximum weight (see

[Mut94, Section 6.1] for details):

4.1
$$\min \sum_{e \in E} w_e x_e \tag{4.1a}$$
$$x(K) \geq 1 \qquad \forall K \in \mathcal{K}_G \tag{4.1b}$$
$$x_e \in \{0,1\} \qquad \forall e \in E \tag{4.1c}$$

Here, we denote $x(K) := \sum_{e \in K} x_e$.

There exist implementations based on this formulation, *e.g.*, provided by the OGDF [Chi+13]. However, this implementation more general for maximum c-planar subgraphs of clustered graphs [FCE95] and therefore not efficient for our usecase. We will use a lightweight self implemented version to compare the different exact algorithms against the formulation (4.1) above. However, we will also compare our own implementation against the implementation provided by the OGDF.

Implementation Details. Our ILP-based approach uses SCIP [Mah+17, version 4.0.0] and OGDF [Chi+13, snapshot 2016.12], both using Gurobi [Gur16, version 7.0.2] as LP solver. Note that in general there is an exponential number of Kuratowski subdivisions in a non-planar graph. Thus we cannot consider all constraints (4.1b) immediately. We use a cutting-plane method and start with no constraint of type (4.1b) and successively add new constraints based on the current LP solution. This method is realizes as constraint handler (SCIP plugin). An implementation of the Boyer-Myrvold planarity test is used to extract multiple Kuratowski subdivisions at once [CMS07]. A top killing heap assures that only the most violating constraints are added to the LP in the Branch&Bound tree. For the root node and in the Branch&Bound process we use an MPS heuristic to improve the bounds. All parameters to control the implementation details are explained below.

Definition (top-k killing heap). A binary heap that holds the k elements with the highest keys and discards the other is called a *top-k killing heap.* ⌟

The implementation is parameterized by the following five options:

thres: A sequence $(t_1, \ldots, t_n)$ of thresholds that is used to transform an LP solution into an ILP solution: A solution $\tilde{x}$ for the relaxation of the current problem is mapped to an integer solution x w.r.t. a threshold t by $x_e = 1$ if and only if $\tilde{x}_e \geq 1 - t$.

Based on this transformation the constraint handler checks if the induced ILP solution is feasible or if there is a Kuratowski subdivision that has to be added. The graph (of edges e with $x_e = 0$) that is induced by the current LP solution is checked by the Boyer-Myrvold planarity test for Kuratowski subdivisions. If the graph is non-planar the used implementation gives a possibly large list of subdivisions. A size-bounded killing heap is used to filter the result set to the most violated constraints. If the heap is not filled by the constraints that are extracted using t_1 the same process starts with t_2 etc. until the heap is full. We tested the parameters (0.1), (0.5), $(0.01, 0.1)$, $(0.1, 0.5)$ and $(0.01, 0.1, 0.5)$.

heap: The maximum size of the killing heap (as described above). It also controls the maximum number of extracted subdivisions by the planarity test. At most $5 \times$ *heap* subdivisions are returned. We tested the parameters 10 and 50.

hcall: An MPS heuristic is used in the Branch&Bound process to find feasible planar subgraphs (induced by the current LP solution). Possible parameters are: The heuristic is called

$hcall = 0$: never

$hcall = 1$: only in the root node as initial feasible solution

hcall = 2: in the root node and for each new Branch&Bound node

hcall = 3: in the root node and after each new LP solution

hrand: Randomization factor *hrand* between 0 and 1 of the heuristic: Let w be the weights of the edges of the input graph. By $\hat{w}$ we denote the weights normalizes to $[0, 1]$. The randomized weights w_r are defined as $w_r(e) = (1 - \mathit{hrand})\hat{w}(e) + \mathit{hrand} \cdot X(e)$, where $X(e)$ is a real random value in $[0, 1]$ for each edge e.

We tested 0.0 and 0.5 for the randomization factor *hrand*.

hruns: Randomized heuristics (with $\mathit{hrand} > 0$) have multiple runs where the final result is the subgraph with the largest weight over all runs. We tested 1 and 5.

The used heuristic itself is also a free parameter but we did not compare different heuristics within our B&B algorithm. A recent study shows that the greedy Cactus algorithm [Căl+98] (extended by a naive approach to maximize the solution) yields the best solutions and has a good running time on real world graphs, see [CKW16, Observation F1 and Figure 1(c)]. The implementation is given by `MaximalPlanarSubgraphSimple` and `PlanarSubgraphCactus` as part of the OGDF.

Optimal Parameter Setting. Let now $U(p)$ denote the set of used test values for a given parameter p. To optimize the parameter setting of the Kuratowski-based ILP we run all 90 variants $\mathcal{S}$:

$$
\begin{aligned}
U(\mathit{thres}) &= \{\mathit{thres}{=}0.1,\quad \mathit{thres}{=}0.5,\quad \mathit{thres}{=}0.01, 0.1,\quad \mathit{thres}{=}0.1, 0.5,\quad \mathit{thres}{=}0.01, 0.1, 0.5\}\\
U(\mathit{heap}) &= \{\mathit{heap}{=}10,\quad \mathit{heap}{=}50\}\\
U(\mathit{hcall}) &= \{\mathit{hcall}{=}0,\quad \mathit{hcall}{=}1,\quad \mathit{hcall}{=}2,\quad \mathit{hcall}{=}3\}\\
U(\mathit{hrand}) &= \{\mathit{hrand}{=}0.0,\quad \mathit{hrand}{=}0.5\}\\
U(\mathit{hruns}) &= \{\mathit{hruns}{=}1,\quad \mathit{hruns}{=}5\}\\
\mathcal{S} &:= U(\mathit{thres}) \times U(\mathit{heap}) \times \big(U(\mathit{hcall}) \setminus \{\mathit{hcall}{=}0\}\big) \times\\
&\quad \big(U(\mathit{hrand}) \times U(\mathit{hruns}) \setminus \{(\mathit{hrand}{=}0.0, \mathit{hruns}{=}5)\}\big)
\end{aligned}
$$

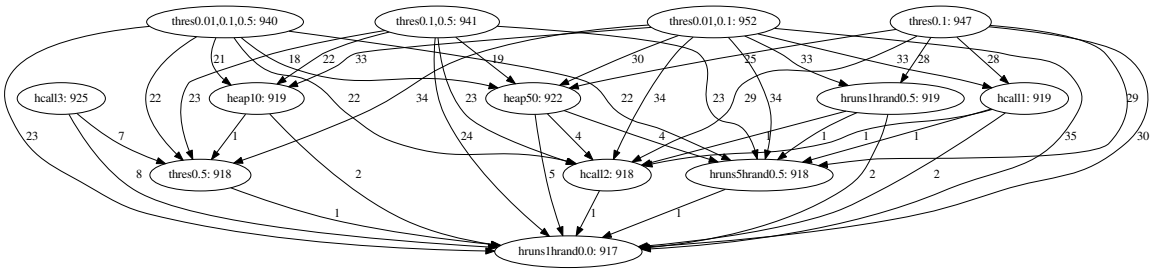
on a random sample of 802 non-planar Rome graphs and all non-planar North graphs. Our C++ code is compiled with GCC 5.3.1-13, and runs on a single core of an Intel(R) Xeon(R) CPU E5-2430 v2 with DDR3 Memory @ 1600 MHz under Debian 8. We applied a memory limit of 8 GB and a time limit of 20 minutes.

A *setting* for our own implementation is $s \in \mathcal{S}$. Let $\mathcal{U}$ denote $U(\mathit{thres}) \cup U(\mathit{heap}) \cup U(\mathit{hcall}) \cup U(\mathit{hrand}) \cup U(\mathit{hruns})$. By $I(s)$ we denote the solved instances with setting s. Using this notation we investigate two different methods to determine a good parameter choice for our own implementation.

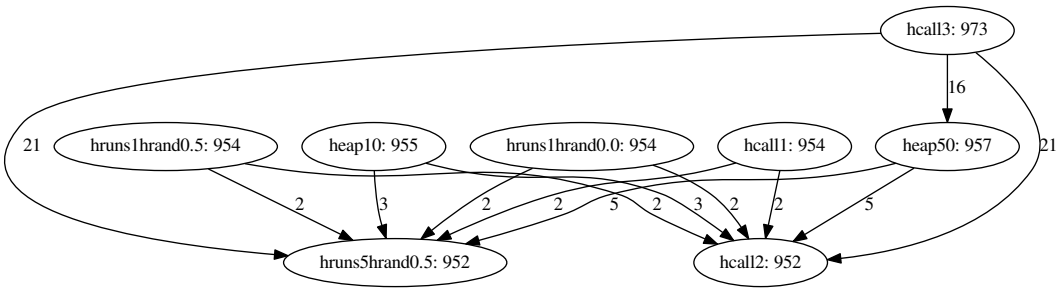
- *Greedy-Max-Solved*: The best setting s^{GMS} is defined as $s^{\mathrm{GMS}} := \arg\max\{|I(s)| : s \in \mathcal{S}\}$.

  This method to determine good parameters for our implementation simply searches for the setting $s \in \mathcal{S}$ with the highest number of solved instances.

- *Successive-Stable-Choice*: Let $u \in \mathcal{U}$ be an arbitrary value of one of the parameters and $S_u = \{s \in \mathcal{S} \mid u \in s\}$ denote the set of all setting where u is part of the setting. We are interested in the intersection of the solved instances over all settings in S_u, *i.e.* $I(u) := \bigcap_{s \in S_u} I(s)$. A *stable* value u^* is defined as $u^* := \arg\max\{|I(u)| : u \in \mathcal{U}\}$. The best setting s^{SSC} is then build by successively picking a stable value, fixing the corresponding parameter to this value and deleting all values for this parameter from $\mathcal{U}$ until all parameters are fixed.



4.A **Figure 4.A:** The graph of all $|I(u)|$ values for $u \in \mathcal{U}$. An arc $u \to u'$ is drawn if $I(u') \subsetneq I(u)$.



4.B **Figure 4.B:** The graph of all $|I(u)|$ values for $u \in \mathcal{U}_1$.

This method to determine good parameters for our implementation examines if the choice of a specific parameter is stable. A value $u \in \mathcal{U}$ for a single parameter is stable if the set of common solved instances (by all parameter settings $u \in s \in \mathcal{S}$) is maximum. This way we answer the question *what is the minimal number of solved instances over all settings when we fix a parameter to a value* $u \in \mathcal{U}$.

On our test set with 1224, instances the best settings according to GMS are

$$s_1^{\text{GMS}} := (\textit{thres}{=}0.01{,}0.1,\ \textit{heap}{=}50,\ \textit{hruns}{=}1,\ \textit{hrand}{=}0.5,\ \textit{hcall}{=}3)$$

and

$$s_2^{\text{GMS}} := (\textit{thres}{=}0.1,\ \textit{heap}{=}10,\ \textit{hruns}{=}1,\ \textit{hrand}{=}0.5,\ \textit{hcall}{=}3)$$

which both solve 988 instances. The SSC method to determine the best setting results in $s^{\text{SSC}} := (\textit{thres}{=}0.01{,}0.1,\ \textit{heap}{=}50,\ \textit{hruns}{=}1,\ \textit{hrand}{=}0.5,\ \textit{hcall}{=}3)$: The sizes of all $I(u)$ for $u \in \mathcal{U}$ are shown in Figure 4.A: The parameter "*thres*=0.01,0.1" is stable. To determine the next stable parameter we focus on $\mathcal{U}_1 := \mathcal{U} \backslash U(\textit{thres})$ on the settings $\mathcal{S}_1 := \{s \in \mathcal{S} : \textit{thres}{=}0.01{,}0.1 \in s\}$. The corresponding graphs is shown in Figure 4.B. On this restricted set a stable parameter is "$\textit{hcall} = 3$". The process continues with Figure 4.C where "$\textit{hruns}{=}1, \textit{hrand}{=}0.5$" is stable. Finally, we only have to consider the parameter *heap* in Figure 4.D, where $\textit{heap} = 50$ is stable.

Thus, in further experiments—where we compare different exact MPS algorithms—we always use $s^{\text{SSC}} = s_1^{\text{GMS}}$ as our parameter setting for the ILP formulation based on Kuratowski subdivisions.

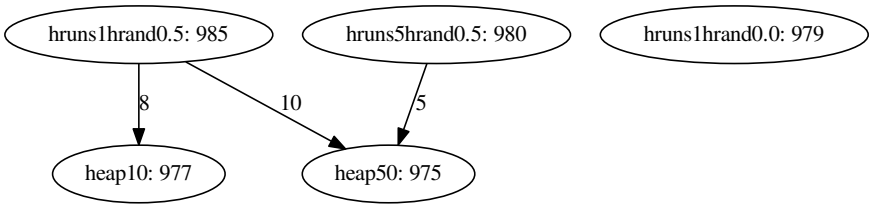


Figure 4.C: The graph of remaining $|I(u)|$ values after fixing *thres*=0.01,0.1 and *hcall* = 3. 4.C

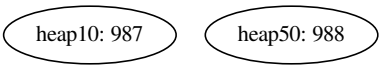


Figure 4.D: The graph of remaining $|I(u)|$ values after fixing *thres*=0.01,0.1, *hcall* = 3, *hruns*=1, and *hrand*=0.5. 4.D

A PBS Formulation via Kuratowski Subdivisions. Also as a reference to compare further exact algorithms we implemented a PBS-based method using Kuratowski subdivisions. We use the PBS solver Clasp 3.2.1 [Geb+11].

In the beginning no constraints of type (4.1b) are used and the graph is checked for planarity. Using the same OGDF routines as before we add multiple Kuratowski constraints to the current PBS formulation until we reach a planar subgraph. Here we use Clasp's ability to perform warm-starts.

4.3 Stronger Formulations using Additional Minors

Lipton et al. [Lip+16] consider seven generalizations of *almost planar graphs*. They consider the question whether the properties are closed under taking minors. Furthermore, they (try to) construct the *Kuratowski sets* for the minor closed properties.

Definition (minor minimal). We say that G is *minor minimal* $\mathcal{P}$ if G has property $\mathcal{P}$ but no proper minor does. ⌟

Lemma. [Lip+16, Corollary 1.3 and 1.4] *For any graph property $\mathcal{P}$, there is a corresponding finite set of minor minimal $\mathcal{P}$ graphs. Let $\mathcal{P}_c$ be graph property that is closed under taking minors. Then there is a finite set of minimal non-$\mathcal{P}_c$ graphs S such that for any graph G, G satisfies $\mathcal{P}_c$ if and only if G has no minor in S.*

Definition (Kuratowski set). When $\mathcal{P}$ is minor closed, we say that the set S from above is the *Kuratowski set* for $\mathcal{P}$. ⌟

Example. $\{K_5, K_{3,3}\}$ is the Kuratowski set for planarity. ⌟

The generalizations considered by Lipton et al. are:

Definition. A planar graph is

(a) *almost non-planar* (AN) if there exist two non-adjacent vertices such that adding an edge between the vertices yields a non-planar graph.

(b) *completely almost non-planar* (CAN) if it is not complete and adding an edge between any pair of non-adjacent vertices yields a non-planar graph.

A graph is

(c) *not apex* (NA) if, for all vertices v, $G - v$ is non-planar.

(d) *not edge apex* (NE) if, for all edges e, $G - e$ is non-planar.

(e) *not contraction apex* (NC) if, for all edges e, G/e is non-planar.

(f) *in-completely apex* (IA) if there is a vertex v such that $G - v$ is non-planar.

(g) *in-completely edge apex* (IE) if there is an edge e such that $G - e$ is non-planar.

(h) *in-completely contraction apex* (IC) if there is an edge e such that G/e is non-planar. ⌟

Some of the generalizations above are of interest for our problem. [Lip+16] contains all minor minimal IA, IE, and some minor minimal NE and NC graphs. This approach can be used to consider a larger set than $\mathcal{K}_G$ when we add further constraints in the B&B process.

Example. Let G be an NE graph. Deleting any edge results in a non-planar graph. Thus, we have to delete at least two edges from G to obtain a planar graph. ⌟

Assume that $\mathcal{M}_G$ is a set of graphs satisfying NE, then (4.1) can be strengthen to:

4.2
$$\min \sum_{e \in E} w_e x_e \tag{4.2a}$$
$$x(K) \geq 1 \qquad \forall K \in \mathcal{K}_G \tag{4.2b}$$
$$x(M) \geq 2 \qquad \forall M \in \mathcal{M}_G \tag{4.2c}$$
$$x_e \in \{0, 1\} \qquad \forall e \in E \tag{4.2d}$$

It remains an open task to identify suitable graphs for $\mathcal{M}_G$ based on [Lip+16]. Furthermore, it is questionable if (4.2) is stronger than (4.1), both in theory and practice.

4.4 Planar Graphs and Total Orders

Based on the planarity criterion by Schnyder [Sch89] we develop in this section an exact algorithm that does not calculate a rotation system itself, but an embedding with exponentially large coordinates. We omit the details for the coordinates, as they are presented in [Sch89, Section 4]. Only the edges of the solution are of interest for us.

Definition (poset). A *partially ordered set*, in short *poset*, is a pair $(S, \prec)$ of a set S together with strict partial order (transitive, irreflexive, binary relation) $\prec$. ⌟

According to [Szp30], every poset $(S, \prec)$ has a set $\mathcal{R}$ of total orders (transitive, antisymmetric, total, binary relation) on S whose intersection is $\prec$. This means that $x \prec y$ if and only if $x <_i y$ for all $<_i \in \mathcal{R}$.

Definition (realizer, Dushnik-Miller dimension). [DM41] A set of total orders whose intersection is a given poset P is called a *realizer* of P. The *Dushnik-Miller dimension* $\dim P$ of P is the minimum cardinality over all realizers of P. ⌟

Let $G = (V, E)$ be a graph and $\mathcal{VE} := V \cup E$. We associate a poset $P_G = (\mathcal{VE}, \prec_G)$ to G by

$$x \prec_G y \qquad \text{if and only if} \qquad y = (v, w) \in E \text{ and } x \in \{v, w\}.$$

The dimension of G is defined as the Dushnik-Miller dimension of P_G. The planarity criterion using total orders is now given by the following theorem.

Theorem. [Sch89, Theorem 4.1 and 6.2] *A graph is planar if and only if its dimension is at most three.*

Remark. [Sch89, Example 2.3] A graph with dimension two is a path. A graph with dimension one is an isolated vertex. Therefore, we will restrict our exact algorithm to check for dimension three. ⌟

The formulation corresponding to the theorem above can be described as follows. Let s_e for $e \in E$ denote if e is in the solution (part of the MPS). The binary variables $p_{x,y}$ denote if $x \prec_G y$. We use $t^i_{x,y}$, for $i \in [3]$, to check if we can build the total orders $<_i$ whose intersection is $\prec_G$.

$$\begin{aligned}
\max \quad & \sum\nolimits_{e \in E} w[e] s_e && && 4.3\\
\text{s.t.} \quad & p_{v,e} = s_e && \forall e \in E,\ \forall v \in e && (4.3a)\\
& p_{v,e} = 0 && \forall e \in E,\ \forall v \in V\colon\ v \notin e && (4.3b)\\
& p_{x,y} = 0 && \forall (x, y) \in (E \times E) \cup (\mathcal{VE} \times V) && (4.3c)\\
& p_{x,y} \le t^i_{x,y} && \forall i \in [3],\ \forall x, y \in \mathcal{VE} && (4.3d)\\
& p_{x,y} \ge t^0_{x,y} + t^1_{x,y} + t^2_{x,y} - 2 && \forall x, y \in \mathcal{VE} && (4.3e)\\
& t^i_{x,y} + t^i_{y,z} - 1 \le t^i_{x,z} && \forall i \in [3],\ \forall \text{ pairwise distinct } x, y, z \in \mathcal{VE} && (4.3f)\\
& t^i_{x,y} + t^i_{y,x} = 1 && \forall i \in [3],\ \forall x \neq y \in \mathcal{VE} && (4.3g)\\
& s_e, p_{x,y}, t^i_{x,y} \in \{0, 1\} && \forall i \in [3],\ \forall e \in E,\ \forall x, y \in \mathcal{VE}
\end{aligned}$$

Theorem 4.4. *Formulation* (4.3) *solves MPS on graphs that are not a path or an isolated vertex.* 4.4

Proof. The idea is to represent a total order on $\mathcal{VE}$ but only ensure that their intersection restricted to the set $V \cup \{e \in E : s_e = 1\}$ if the order $\prec_H$ for $H = G[\{e \in E : s_e = 1\}]$.

Here Equations (4.3a)–(4.3c) ensure that p represents $\prec_G$ on the subgraph induced by s. The equality $\prec_G = <_0 \cap <_1 \cap <_2$ is realized by Equation (4.3d) and Equation (4.3e). We only realize $\prec_H$ described above as follows: Let $S' := \{e \in E : s_e = 0\}$ be the edges that are not in H. All elements of S' occur in the three total orders. To be consistent with (4.3a) the elements $s_1, \ldots, s_{|S'|}$ of S' have to occur as $s_1 <_i s_2 <_i \cdots <_i s_{|S'|}$ and $s_{|S'|} <_j s_{|S'|-1} <_j \cdots <_j s_1$ in two orders i and j. Thus, for any possible set S of edges in the optimum H we find total orders consistent with $\prec_H = <_0 \cap <_1 \cap <_2$ where the right term is restricted to $V \cup S$.

Finally, each t^i has to be a total order: transitive by Equation (4.3f), antisymmetric and total by Equation (4.3g). Thus, the formulation yields an MPS of G by searching for the largest subgraph of dimension three. □

The ILP above only implements the basic idea of the presented criterion. We use the following result to speed-up the exact algorithm:

Lemma 4.5. [Sch89, Lemma 2.1] *A graph $G = (V, E)$ has dimension at most d if and only if there exists a sequence $<_1, \ldots, <_d$ of total orders on V such that* 4.5

1. *the intersection of* $<_1, \ldots, <_d$ *is empty, and*
2. *for each edge* $\{x, y\} \in E$ *and each vertex* $z \notin \{x, y\}$ *of* G*, there is at least one order* $<_i$ *such that* $x <_i z$ *and* $y <_i z$*.*

The formulation corresponding to the lemma above only uses the $t^i_{x,y}$ variables restricted to V together with the s_e variables.

4.6

$$
\begin{aligned}
\max \quad & \sum\nolimits_{e \in E} w[e] s_e & & \\
\text{s.t.} \quad & s_e \leq a^1_{e,z} + a^2_{e,z} + a^3_{e,z} & \forall e = \{x, y\} \in E, \forall z \in V\colon z \notin e & \text{(4.6a)} \\
& a^i_{e,z} \geq t^i_{x,z} + t^i_{y,z} - 1 & \forall i \in [3], \forall e = \{x, y\} \in E, \forall z \in V\colon z \notin e & \text{(4.6b)} \\
& a^i_{e,z} \leq t^i_{x,z} & \forall i \in [3], \forall e = \{x, y\} \in E, \forall z \in V\colon z \notin e & \text{(4.6c)} \\
& a^i_{e,z} \leq t^i_{y,z} & \forall i \in [3], \forall e = \{x, y\} \in E, \forall z \in V\colon z \notin e & \text{(4.6d)} \\
& t^0_{x,y} + t^1_{x,y} + t^2_{x,y} \leq 2 & \forall x, y \in V & \text{(4.6e)} \\
& t^i_{x,y} + t^i_{y,z} - 1 \leq t^i_{x,z} & \forall i \in [3], \forall \text{ pairwise distinct } x, y, z \in V & \text{(4.6f)} \\
& t^i_{x,y} + t^i_{y,x} = 1 & \forall i \in [3], \forall x \neq y \in V & \text{(4.6g)} \\
& s_e, a^i_{e,z}, t^i_{x,y} \in \{0, 1\} & \forall i \in [3], \forall e \in E, \forall x, y, z \in V &
\end{aligned}
$$

We need auxiliary a-variables to describe $a^i_{\{x,y\},z} \equiv t^i_{x,z} \wedge t^i_{y,z}$, which is realized by (4.6b), (4.6c) and (4.6d). Equation (4.6e) checks if the intersection of the total orders is empty. As in the previous formulation, we need (4.6f) and (4.6g) to ensure that t^i represents a total order. Finally, the constraint (4.6a) equals the second part of Lemma 4.5.

Remark. Note that Inequality (4.6b) is not necessary, since the objective function together with Inequality (4.6a) will set a-variables to one if possible. ⌟

Remark. To compare the formulations with and without explicit p-variables we use the same sample of 802 Rome graphs as in Section 4.2 and work on the same hardware. Using formulation (4.3) with p-variables we solve 84 instances. With formulation (4.6) we solve 123 instances. The set of instances solved by (4.3) is a subset of the instances solved by (4.6). ⌟

The variant where the total orders are restricted to V is very promising for a pseudo-Boolean SAT formulation since the relaxation of (4.6) is obviously very weak. Our PBS formulation uses the same variables as above.

$$
\begin{aligned}
\max \quad & \sum\nolimits_{e \in E} w[e] s_e & & \\
\text{s.t.} \quad & s_e \Leftrightarrow a^1_{e,z} \vee a^2_{e,z} \vee a^3_{e,z} & \forall e = \{x, y\} \in E, \forall z \in V\colon z \notin e & \\
& a^i_{e,z} \Leftrightarrow t^i_{x,z} \wedge t^i_{y,z} & \forall i \in [3], \forall e = \{x, y\} \in E, \forall z \in V\colon z \notin e & \\
& \neg t^0_{x,y} \vee \neg t^1_{x,y} \vee \neg t^2_{x,y} & \forall x, y \in V & \\
& t^i_{x,y} + t^i_{y,z} - 1 \leq t^i_{x,z} & \forall i \in [3], \forall \text{ pairwise distinct } x, y, z \in V & \\
& t^i_{x,y} \operatorname{XOR} t^i_{y,x} & \forall i \in [3], \forall x \neq y \in V & \\
& s_e, a^i_{e,z}, t^i_{x,y} \in \{\texttt{true}, \texttt{false}\} & \forall i \in [3], \forall e \in E, \forall x, y, z \in V &
\end{aligned}
$$

The experimental comparison of the ILP and PBS formulation against each other and against the other formulations of this chapter can be found in Section 4.7.

Formulations using Betweenness Variables. To omit the introduction of auxiliary variables $a^i_{e,z}$ to denote the logical conjunction (4.6b)–(4.6d) we represent the total orders via betweenness variables. Caprara et al. [Cap+11] present the usage and impact of such variables to represent total orders.

Notation. Let u, w, x, z be four elements of a set that appear as $u <_i w <_i x <_i z$ in a total order $<_i$. We write $[u, w, x, z]_i$ in short. By $[\underline{u}, \overline{w}, \underline{x}, \overline{z}] \Rightarrow_i [\overline{u}, \underline{w}, \overline{x}, \underline{z}]$ we denote that $u <_i w <_i x$ together with $w <_i x <_i z$ imply that $u <_i w <_i z$ and $u <_i x <_i z$. ⌟

We exchange the total order variables t^i with betweenness variables b^i. For a triple x, y, z of vertices we denote by $b^i_{x,y,z}$ if $x <_i y <_i z$ holds. The formulation using betweenness variables is:

$$\max \quad \sum\nolimits_{e \in E} w[e] s_e \tag{4.7}$$

$$\text{s.t.} \quad s_e \le \sum\nolimits_{i \in [3]} b^i_{x,y,z} + b^i_{y,x,z} \qquad \forall e = \{x, y\} \in E,\ \forall z \in V\colon z \notin e \tag{4.7a}$$

$$\sum\nolimits_{\sigma \in \mathrm{Sym}(S)} b^i_{\sigma_1,\sigma_2,\sigma_3} = 1 \qquad \forall i \in [3],\ \forall S := \{x, y, z\} \subseteq V \tag{4.7b}$$

$$b^i_{u,w,x} + b^i_{u,x,z} - 1 \le b^i_{u,w,z} \qquad \forall i \in [3],\ \forall \text{ p.d. } u, w, x, z \in V \tag{4.7c}$$

$$b^i_{u,w,x} + b^i_{u,x,z} - 1 \le b^i_{w,x,z} \qquad \forall i \in [3],\ \forall \text{ p.d. } u, w, x, z \in V \tag{4.7d}$$

$$b^i_{u,w,x} + b^i_{w,x,z} - 1 \le b^i_{u,w,z} \qquad \forall i \in [3],\ \forall \text{ p.d. } u, w, x, z \in V \tag{4.7e}$$

$$b^i_{u,w,x} + b^i_{w,x,z} - 1 \le b^i_{u,x,z} \qquad \forall i \in [3],\ \forall \text{ p.d. } u, w, x, z \in V \tag{4.7f}$$

$$\sum_{i=1}^{3} \sum_{x \in V \setminus \{u,w\}} b^i_{w,u,x} + b^i_{w,x,u} + b^i_{x,w,u} \ge 1 \qquad \forall \text{ p.d. } u, w \in V \tag{4.7g}$$

$$s_e, b^i_{x,y,z} \in \{0, 1\} \qquad \forall i \in [3],\ \forall e \in E,\ \forall \text{ p.d. } x, y, z \in V$$

For a given edge $e = \{x, y\}$, the requirement that there is an ordering $<_i$ with $x <_i z$ and $y <_i z$ is equivalent to the two possibilities $x <_i y <_i z$ or $y <_i x <_i z$. Thus, Equation (4.7a) is equivalent to the connection of the a- and t-variables in the formulation (4.6). Equation (4.7b) prohibits any symmetries in the b-variables. We ensure that the intersection of the induced total orders is empty by (4.7g): Assume that the intersection of the orders it not empty and we have $u < w$ in the intersection. This can only happen when

$$\sum_{i=1}^{3} \sum_{x \in V \setminus \{u,w\}} b^i_{w,u,x} + b^i_{w,x,u} + b^i_{x,w,u} = 0,$$

as there is no occurrence of a contradicting $w <_i u <_i x$, $w <_i x <_i u$, or $x <_i w <_i u$ in any of the orders $<_i$.

Finally, for four vertices that appear as $[u, w, x, y]$ in an order $<_i$, we have to ensure that $[\underline{u}, \overline{w}, \underline{x}, \overline{z}] \Rightarrow_i [\overline{u}, \underline{w}, \overline{x}, \underline{z}]$ and $[\overline{u}, \underline{w}, \overline{x}, \underline{z}] \Rightarrow_i [\underline{u}, \overline{w}, \overline{x}, \underline{z}]$, using Equations (4.7c)–(4.7f).

Remark. When using the betweenness approach to represent the three total orders the number of solved instances (on our set of 802 Rome graphs) drops to 77. ⌟

Constructing Feasible Solutions from Heuristics. Following the same workflow as for the other exact algorithms we want to construct a feasible solution for our ILP based on a given heuristic. An obvious connection between our task and Schnyder-layouts [Sch90] is given as follows:

In [Sch89, Section 3] the author defines a connection between so-called *three-dimensional representations* $<_1, <_2, <_3$ on V (a sequence of total orders on V that fulfill only part one of

Lemma 4.5) and partial orders $<_1^*, <_2^*, <_3^*$ on V. The input graph $G = (V, E)$ is decomposed into three arc disjoint digraphs with arc sets A_1, A_2, A_3, where $A_i = \big\{(x, y) \mid \{x, y\} \in E \wedge x <_i^* y\big\}$. The details are omitted here. [Sch89, Theorem 3.3] shows that each (V, A_i) is a rooted forest. Such structures are the fundament of Schnyder-layouts.

The OGDF contains an implementation of the algorithm described in [Sch90, Section 8]. However, either the implementation or the trees constructed in [Sch90] itself differ from our required A_i's. One could use Section 5 and 6 from [Sch89] to construct the needed A_i's (and thus the orders $<_1, <_2, <_3$). Because of time limits, we refrain from doing so.

PBS Formulation and Parameter Setting. The ILP formulations presented in this section are easily transformable into PBS formulations. We refrain from explicitly presenting the formulations.

Our evaluation of the different formulations in this section showed that using Lemma 4.5 (compared to representing the partial orders explicitly) and representing total orders in the direct way (not via betweenness variables) achieves the best performance. We will use this parameter setting when comparing the formulation based on total orders against other formulations.

4.5 A Formulation based on Theta Graphs

Archdeacon and Širáň [AŠ98] presented a planarity criterion that checks if all cycles in a auxiliary signed graph contain an even number of edges with a negative sign.

Graphs in this section are simple and undirected. Note that the criterion also works on graphs with multi-edges.

Definition (claw, theta graph). A *claw* in a graph is a set of three pairwise adjacent darts. A claw is *rooted* at the common incident vertex. A *theta graph* is a pair of disjoint vertices joined by three pairwise internally-vertex-disjoint paths. ⌟

A theta graph contains exactly two claws rooted at the two degree-three vertices.

Definition (signature, signed graph, balanced). A *signature* $\lambda\colon E \to \{1, -1\}$ on a graph is an assignment of a plus or minus sign on each edge. A *signed graph* is a graph together with a signature. A cycle in a signed graph is *balanced* if and only if it contains an even number of negative edges. A signed graph is *balanced* if and only if every cycle is balanced. ⌟

Definition (local switch, equivalent). A *local switch* on a signed graph reverses the sign of each edge incident with a given vertex. Two signed graphs are *equivalent* if there is a sequence of local switches transforming one signature into the other. ⌟

Lemma. *A balanced graph is equivalent to one signed with every edge positive.*

Proof. Recall Lemma 2.4. Any signature is equivalent to one where the edges in a fixed spanning tree T are all signed positive. If the edges of T are all positive in a balanced signature, then every edge not in T is also positive: Otherwise, we would have a cycle using one non-tree edge with negative sign and tree-edges with positive signs which contradicts to the assumption that we have a balanced graph. □

There are two embeddings of a theta graph into an oriented plane. These embeddings can be described in terms of local rotations: cyclic permutations of the half-edges incident with a vertex. Beginning with a planar embedding of a theta graph, reversing the local rotation at exactly one degree-three vertex results in a non-planar embedding.

Definition (signed triple graph Ψ_G). The *signed triple graph* $\Psi_G = (C, \Theta)$ of a graph G is defined as: The vertex set C is all claws of G. Arbitrarily fix a local rotation on each claw. Two claws are joined by an edge $\theta \in \Theta$ if they lie in a common theta graph τ. The sign of θ is $+1$ if the local rotations of the two claws embed τ in the plane. Otherwise, it is -1. ⌟

Theorem. [AŠ98, Theorem 2.1] *G is planar if and only if Ψ_G is balanced.*

To use the planarity criterion from above we first describe an exact algorithm that only tests for planarity. Note that this implies a substantial overhead compared to known linear-time planarity tests. We will use this planarity test to build an exact algorithm for MPS.

Identifying Theta Graphs. Let $G = (V, E)$ be a given input graph. The set C of all claws in G can be computed in $\mathcal{O}(|V|^4)$ time. To construct the edges of Ψ_G we have to check if there are three vertex-disjoint paths between pairs of claws. We transform this problem into a max-flow problem in the *split graphs*: Pick two claws with roots r_1 and r_2. Split every vertex that is not r_i into two vertices connected by an edge each. A set of three vertex-disjoint paths from r_1 to r_2 in G corresponds to a flow with value three in the split graph with unit capacities.

Let c_1 and c_2 be nodes in Ψ_G (c_1 and c_2 is a pair of claws) with roots r_1 and r_2. They lie in a common theta graph if there is a r_1-r_2-flow of value 3 in the following auxiliary directed graph $G^{\text{split}}_{c_1,c_2} = (V^{\text{s}}, A^{\text{s}})$ with unit capacities:

Let $V' := V \setminus \{r_1, r_2\}$. Each node of $v \in V'$ is split into v_1 and v_2. We define $V^{\text{s}} := \{r_1, r_2\} \cup \{v_1, v_2 : v \in V'\}$. Let $N'(v)$ denote the neighbors of v in $G[V']$. For each claw c_i we denote its vertices with $V(c_i) = \{r_i, v_i^1, v_i^2, v_i^3\}$, where r_i is the root of c_i. In short, we write $V_{c_i} := V(c_i) \setminus \{r_1\}$. The dart set of $G^{\text{split}}_{c_1,c_2}$ is

$$A^{\text{s}} := \{r_1 \to v : v \in V_{c_1}\} \cup \{v \to r_2 : v \in V_{c_2}\} \cup \bigcup_{v \in V'} \{u \to v_1 : u \in N'(v)\} \;\cup\; \{v_1 \to v_2 : v \in V'\} \;\cup\; \bigcup_{v \in V'} \{v_2 \to u : u \in N'(v)\}.$$

Clearly, a 0/1-flow in $G^{\text{split}}_{c_1,c_2}$ from r_1 to r_2 with value 3 is equivalent to a set of three internally vertex-disjoint r_1-r_2-paths: A flow of value 3 corresponds to three paths. The paths are internally vertex-disjoint because we have unit capacities on the darts $\{v_1 \to v_2 : v \in V'\}$.

For each claw pair we can check if there is an edge in Ψ_G by using, *e.g.*, the Ford-Fulkerson algorithm [FF56] to solve the flow problem above in $\mathcal{O}(|A^{\text{s}}|) = \mathcal{O}(|E|)$ time. The graph Ψ_G can thus be computed in $\mathcal{O}(|V|^8|E|)$ time.

Testing Planarity Using Theta Graphs. As in Section 4.4 we use betweenness variables to describe the rotation system of the input graph G. The basic idea of this planarity test is to ensure that in each theta graph the betweenness variables of the two corresponding claws are consistent. If there is a rotation system for G such that the local rotations between all theta graphs are consistent we have an embedding of G in the plane.

We use variables $b^v_{x,y,z}$ for each triple of pairwise different $x, y, z \in N(v)$. By $b^v_{x,y,z} = 1$ we denote that y is somewhere between x and z in the counter-clockwise rotation at v. For each $\theta \in \Theta$ we write θ_1 and θ_2 for the two corresponding claws.

The ILP realizing the planarity test is:

4.8
$$\max \quad 1 \tag{4.8a}$$
$$\text{s.t.} \quad b^v_{x,y,z} = b^v_{y,z,x} = b^v_{z,x,y} = 1 - b^v_{x,z,y} \qquad \forall v \in V, \text{ p.d. } x,y,z \in N(v) \tag{4.8b}$$
$$b^v_{u,w,x} + b^v_{u,x,z} - 1 \leq b^v_{u,w,z} \qquad \forall v \in V, \forall \text{ p.d. } u,w,x,z \in N(v) \tag{4.8c}$$
$$b^v_{u,w,x} + b^v_{u,x,z} - 1 \leq b^v_{w,x,z} \qquad \forall v \in V, \forall \text{ p.d. } u,w,x,z \in N(v) \tag{4.8d}$$
$$b^v_{u,w,x} + b^v_{w,x,z} - 1 \leq b^v_{u,w,z} \qquad \forall v \in V, \forall \text{ p.d. } u,w,x,z \in N(v) \tag{4.8e}$$
$$b^v_{u,w,x} + b^v_{w,x,z} - 1 \leq b^v_{u,x,z} \qquad \forall v \in V, \forall \text{ p.d. } u,w,x,z \in N(v) \tag{4.8f}$$
$$b^{r_1}_{v_1^1,v_1^2,v_1^3} = b^{r_2}_{v_2^1,v_2^2,v_2^3} \qquad \forall \theta = \theta_1\theta_2 \in \Theta \tag{4.8g}$$
$$b^v_{x,y,z} \in \{0,1\} \qquad \forall v \in V, \text{ p.d. } x,y,z \in N(v)$$

4.9 **Theorem 4.9.** *Formulation* (4.8) *is a planarity test.*

Proof. The constraints (4.8b) realize the symmetries within each local rotation. Furthermore, (4.8c)–(4.8f) are the betweenness constraints we already know from Section 4.4. Finally, Equation (4.8g) ensures that the local rotations of both claws of each theta graph are consistent. If there is a feasible solution for (4.8), each edge in Ψ_G can be signed with $+1$ and thus Ψ_G is balanced. □

Solving MPS Using Theta Graphs. The formulation above can be extended to solve MPS. For this, we introduce new variables s_e for $e \in E$ to denote if an edge e is in the solution ($s_e = 1$) or not ($s_e = 0$). To find a maximum planar subgraph H of G we maximize over $\sum_{e \in E} s_e$ where the constraints of our MPS formulation are:

- A claw $c \in C$ is *active in* H if $\sum_{e \in E(c)} s_e = 3$.
- A theta graph $\theta = \theta_1\theta_2 \in \Theta$ is *active in* H, if both of its claws θ_1 and θ_2 are active.
- An active theta graph is in Ψ_H if there is a flow with value 3 in the extended version of the flow problem above:

  The vertices V^{s} and darts A^{s} of the auxiliary graph are constructed in the same way as before. But instead of using unit capacities, the capacities of darts that correspond to edges in G have capacity s_e and the remaining darts have capacity one.
- We reuse the betweenness variables $b^v_{x,y,z}$ to represent the rotations around each vertex v. Again, constraints (4.8b)–(4.8f) ensure that we have feasible rotation system. Note that we include neighbors u of a vertex v in the rotation around v even if $s_{v \to u} = 0$. The formulation can be readjusted to only build rotations for neighbors that are adjacent to v in the solution. But it is also possible to assign them a place in the rotation around v even we do not use the corresponding edge in the solution.
- A theta graph is called *responsible* if it is an active theta graph in Ψ_G and it remains an active theta graph in Ψ_H, *i.e.*, it is active in Ψ_G and there is a flow of value 3 in the extended flow formulation.

  We adapt constraint (4.8g) such that it only ensures that the local rotations of claws in responsible theta graphs are consistent.

Due to the complexity of the arising formulation we refrain from explicitly giving the formulation of the exact algorithm based on theta graphs. For the same reason we did not implement this algorithm.

Note that the presented ILP-based algorithm can easily transformed into a SAT- or PBS-based algorithm.

4.6 Euler Characteristic and Simulated Facial Walks

In [Bey+16] we developed exact algorithms for the Minimum Genus Problem. The idea is to find an embedding with a maximum number of faces. Using the Euler characteristic $|V| - |E| + |F| = \chi = 2 - 2g$, where $|F|$ is the number of faces and g is the resulting (oriented) genus.

A similar idea is the fundament for the exact algorithm that we present in this section. A maximum planar subgraph H obviously uses all nodes V of the input graph $G = (V, E)$: The solution is always a spanning subgraph of the connected input graph. When we search for a solution H we look for genus zero subgraphs. Using the Euler characteristic we get $|V| - |E(H)| + |F(H)| = 2$, or equivalently

$$|E(H)| = |F(H)| + |V| - 2 = |F(H)| + \text{const.}$$

It follows that searching for a planar subgraph with maximal edge cardinality is equivalent to the search for a subgraph with maximal edge cardinality that satisfies the Euler characteristic for genus zero. We therefore simulate the face tracing algorithm with our exact algorithm.

Recall the definitions of facial walks and the face traversal procedure for oriented surfaces from Section 1.1.3.

For this section we always assume that $G = (V, E)$ is a simple, undirected and connected graph with edge weights $w \colon E \to \mathbb{N}$.

4.6.1 Exponentially Sized Formulations: Basic ILP and SAT Models

Our formulation is based on finding an embedding with the largest number of faces. It statically simulates the face tracing algorithm. Note that the solution H can have nodes of degree one.

Let $\bar{f}$ be an upper bound on the attainable number of faces; see Section 2.2 on how to obtain a simple linear bound. For each $i \in [\bar{f}]$, we have a binary variable x_i that is 1 if and only if the i-th face exists and a binary variable c^i_a, for each $a \in A$, that is 1 if and only if arc a is traversed by the i-th face. For each vertex $v \in V$ and neighbors $u, w \in N(v)$, the binary variable $p^v_{u,w}$ is 1 if and only if w is the direct successor of u in the rotation at v in the solution. Note that a feasible assignment of the p^v variables correspond to a rotation around v that only considers neighbors that are adjacent within the solution H. s_e denotes if the edge e is in the solution H. For an arc $a \in A$, s_a equals s_e for the corresponding edge e of a.

We define the following short-hand notations:

$$p^v(I \times J) := \sum_{i \in I} \sum_{j \in J} p^v_{i,j}, \qquad s(I) := \sum_{i \in I} s_i,$$
$$x(I) := \sum_{i \in I} x_i, \qquad s_v(I), := s(\{v\} \times I),$$
$$c^I(J) := \sum_{i \in I} \sum_{j \in J} c^i_j.$$

Using the variables and notations defined above our first ILP formulation using facial walks

is then:

4.10
$$\max \quad \sum_{e\in E} w_e s_e \tag{4.10a}$$
$$\text{s.t.} \quad x([\bar{f}]) = 2 - n + s(E) \tag{4.10b}$$
$$x_1 = 1 \tag{4.10c}$$
$$x_i \geq x_{i+1} \qquad i = 0,\dots,\bar{f}-1 \tag{4.10d}$$
$$x_i \leq c^i(A)/3 \qquad \forall i \in [\bar{f}] \tag{4.10e}$$
$$c^i_a \leq x_i \qquad \forall a \in A,\ i \in [\bar{f}] \tag{4.10f}$$
$$c^{[\bar{f}]}(a) = s_a \qquad \forall a \in A \tag{4.10g}$$
$$c^i(\delta^-(v)) = c^i(\delta^+(v)) \qquad \forall i \in [\bar{f}],\ v \in V \tag{4.10h}$$
$$c^i_{vw} \geq c^i_{uv} + p^v_{u,w} - 1 \qquad \forall i \in [\bar{f}],\ v \in V,\ u,w \in N(v) \tag{4.10i}$$
$$c^i_{uv} \geq c^i_{vw} + p^v_{u,w} - 1 \qquad \forall i \in [\bar{f}],\ v \in V,\ u,w \in N(v) \tag{4.10j}$$
$$p^v(u \times N(v)) = s_{vu} \qquad \forall vu \in A \tag{4.10k}$$
$$p^v(N(v) \times w) = s_{vw} \qquad \forall vw \in A \tag{4.10l}$$
$$p^v(U \times N(v)\backslash U) \geq s_v(\{u,\tilde{u}\}) - 1 \qquad \forall v \in V,\ \emptyset \neq U \subsetneq N(v),\ u \in U,\ \tilde{u} \in N(v)\backslash U \tag{4.10m}$$
$$p^v_{u,u} + s_{vw} \leq 1 \qquad \forall v \in V,\ u \neq w \in N(v) \tag{4.10n}$$
$$s_e, x_i, c^i_a, p^v_{u,w} \in \{0,1\} \qquad \forall e \in E,\ i \in [\bar{f}],\ a \in A,\ v \in V,\ u,w \in N(v)$$

Theorem. *Formulation* (4.10) *solves MPS.*

Proof. The input for the ILP is a weighted simple graph, since it is a non-trivial non-planar-core of a biconnected component. Thus, we maximize in (4.10a) the weights of the selected edges in the solution. The Euler characteristic in equation (4.10b) for genus 0 ensures that the set $\{e : s_e = 1\}$ of selected edges resembles a planar graph. W.l.o.g. we have a spanning tree as a feasible solution, thus we have at least one face, see (4.10c). Inequalities (4.10d) are just symmetry breaking constraints for the x variables. Constraints (4.10e) ensure that if a face exists, it traverses *at least* three arcs; inversely, each arc is traversed by exactly one face due to (4.10g). An arc can only be used in a face, if the face itself is in the solution, which is realized by (4.10f). Equalities (4.10h) guarantee that at every vertex of a face i, the number of i-traversed incoming and outgoing arcs is identical. Inequalities (4.10i) and (4.10j) ensure that arcs uv and vw are both in the same face if w is the successor of u in the rotation at v. Constraints (4.10k) and (4.10l) ensure that p^v represents a permutation of the vertices in $N(v)$ that are incident to v in the solution. The permutation itself has to be a single cycle over the neighbors that are incident to v in the solution. This is achieved by CUT constraints (4.10m) over all subsets of the neighborhood of each node. In contrast to the formulations for the Minimum Genus Problem, we have to deal with degree-one nodes in the solution. If a node u is its own successor in the rotation p^v, the node v is a degree-one node in the solution and thus it cannot have additional outgoing edges. This is realized by (4.10n). □

Some constraints in our initial formulation are redundant:

4.11 **Lemma 4.11.** *Inequalities* (4.10n) *are contained in the CUT constraints* (4.10m) *and* (4.10k).

Proof. We set $U = \{u\}$ in (4.10m) which results in

$$1 \overset{(4.10m)}{\geq} s_v(\{u,\tilde{u}\}) - p^v(U \times N(v) \setminus U) = s_v(\{u,\tilde{u}\}) - p^v(u \times N(v) \setminus \{u\})$$
$$= s_v(\{u,\tilde{u}\}) - (p^v(u \times N(v)) - p^v_{u,u}) = s_{vu} - p^v(u \times N(v)) + s_{v\tilde{u}} + p^v_{u,u} \overset{(4.10k)}{=} s_{v\tilde{u}} + p^v_{u,u}.$$

The resulting inequalities $s_{v\tilde{u}} + p^v_{u,u} \leq 1\ \forall \tilde{u} \in N(v) \setminus U$ are equal to (4.10n). □

Subtour-Elimination Formulation. Inspired by formulations for the k-cardinality tree problem (k-CTP) (see [Chi+09] for an overview of exact ILP-based algorithms using directed cuts) we use a generalization of subtour elimination constraints, introduced by [Fis+94, Sect. 3], to replace the CUT constraints that ensure the permutation p^v around v consists of only one cycle.

The k-CTP is to find a minimum weight tree with exactly k edges in a given undirected edge weighted graph. The formulation based on *general subtour elimination constraints* (GSEC) is simple: The binary variables are z_e and y_v representing if an edge e and a vertex v are part of the solution, respectively. The problem is then described by

$$\begin{aligned} \min \quad & \textstyle\sum_{e\in E} w_e z_e \\ \text{s.t.} \quad & z(E(S\times S)) \le y(S\setminus\{t\}) \quad \forall S\subset V, |S|\ge 2, \forall t\in S && \text{(4.12a)} \\ & z(E) = k \\ & y(V) = k+1 \end{aligned} \qquad 4.12$$

The GSEC shown in (4.12a) ensure that every proper subset of the solution does not contain a subtour. The constraints (4.12a) enforce that every set of $|S| =: \xi + 1$ has at most ξ edges, and therefore contains no cycle.

The formulation using GSEC for our problem is:

$$\begin{aligned} \max \quad & \textstyle\sum_{e\in E} w_e s_e \\ \text{s.t.} \quad & \text{(4.10b)–(4.10l)} \\ & p^v(U\times U) - s_v(U\setminus\{u\}) \le 1 - s_{v\tilde{u}} && \forall v\in V,\ \emptyset\neq U\subsetneq N(v),\ u\in U,\ \tilde{u}\in N(v)\setminus U \quad \text{(4.10m}') \\ & s_e, x_i, c_a^i, p_{u,w}^v \in \{0,1\} && \forall e\in E,\ i\in[\bar{f}],\ a\in A,\ v\in V,\ u,w\in N(v) \end{aligned} \qquad 4.10'$$

The left hand side of (4.10m′) equals (4.12a), but we need a correction term: If there is a neighbor $\tilde{u}$ of v that not in U but the edge $v\tilde{u}$ is part of the solution, we have to forbid that we have a subtour inside U.

Theorem 4.13. *The CUT and GSEC formulations for MPS via facial walks have equally strong LP-relaxations.* 4.13

Proof. Note that

$$p^v(U\times U) = p^v(U\times N(v)) - p^v(U\times N(v)\setminus U)$$

by definition and $s_v(U) = p^v(U\times N(v))$ by (4.10k). Thus,

$$\begin{aligned} & p^v(U\times U) - s_v(U\setminus\{u\}) \le 1 - s_{v\tilde{u}} \\ \Leftrightarrow \quad & [p^v(U\times N(v)) - p^v(U\times N(v)\setminus U)] - [p^v(U\times N(v)) - s_{vu}] \le 1 - s_{v\tilde{u}} \\ \Leftrightarrow \quad & -p^v(U\times N(v)\setminus U) + s_{vu} \le 1 - s_{v\tilde{u}} \end{aligned}$$

for all $v\in V$, $\emptyset\neq U\subsetneq N(v)$, $u\in U$ and $\tilde{u}\in N(v)\setminus U$. Hence, (4.10m) and (4.10m′) are equal.□

Eliminating the Solution-Variables. The variable space can be reduced as the s- and c-variables are connected in the following way

$$s_e = 1 \quad \Leftrightarrow \quad \exists i\in[\bar{f}], \exists a\in A\colon\ E(a) = e \text{ and } c_a^i = 1.$$

We therefore have a formulation using only the face, containment and rotation variables:

4.14
$$\begin{aligned}
\max\quad & \sum\nolimits_{a\in A} w[E(a)]c^{[\bar f]}(a) && && (4.14a)\\
\text{s.t.}\quad & (4.10c)\text{–}(4.10f), (4.10h)\text{–}(4.10j) && &&\\
& 2x([\bar f]) = 4-2n+c^{[\bar f]}(A) && && (4.14b)\\
& c^{[\bar f]}(a) = c^{[\bar f]}(\mathrm{rev}(a)) && \forall a\in A && (4.14c)\\
& c^{[\bar f]}(a) \le 1 && \forall a\in A && (4.14d)\\
& p^v(u\times N(v)) = c^{[\bar f]}(vu) && \forall vu\in A && (4.14e)\\
& p^v(N(v)\times w) = c^{[\bar f]}(vw) && \forall vw\in A && (4.14f)\\
& p^v(U\times \bar U) \ge c^{[\bar f]}(\{vu, v\tilde u\})-1 && \forall v\in V, \forall \emptyset\neq U\subsetneq N(v), \forall u\in U, \forall \tilde u\in \bar U && (4.14g)\\
& x_i, c^i_a, p^v_{u,w} \in \{0,1\} && \forall i\in[\bar f], \forall a\in A, \forall v\in V, \forall u,w\in N(v) &&
\end{aligned}$$

4.15 **Theorem 4.15.** *Let $\mathcal{P}_{(4.10)}$ be the polyhedron corresponding to the CUT LP-relaxation and $\mathcal{P}_{(4.14)}$ the polyhedron corresponding to the LP-relaxation of (4.14), that is,*

$$\begin{aligned}
\mathcal{P}_{(4.10)} &:= \{(s,x,c,p)\in[0,1]^{|E|+\bar f+\bar f\cdot|A|+|V|^3} : (s,x,c,p) \text{ satisfies (4.10)}\},\\
\mathcal{P}_{(4.14)} &:= \{\quad (x,c,p)\in[0,1]^{\bar f+\bar f\cdot|A|+|V|^3} : \qquad (x,c,p) \text{ satisfies (4.14)}\}.
\end{aligned}$$

Both formulations have equally strong LP-relaxations, that is, $\mathcal{P}_{(4.10)} = \mathcal{P}^{+s}_{(4.14)}$, where $\mathcal{P}^{+s}_{(4.14)}$ is the projection of $P_{(4.14)}$ onto the (s,x,c,p) variable space. The x, c and p variables are thereby projected using the identity function and $s_e := c^{[\bar f]}(A(e))/2$ for all $e\in E$.

Proof. We prove equality by showing mutual inclusion:

$\mathcal{P}_{(4.10)} \subseteq \mathcal{P}^{+s}_{(4.14)}$: Let $(\hat s,\hat x,\hat c,\hat p)\in\mathcal{P}_{(4.10)}$ be arbitrary but fixed. Using Equation (4.10g) we see that (4.14e), (4.14f) and (4.14g)—the equivalents of (4.10k), (4.10l) and (4.10m)—are fulfilled because $\hat s_a = \hat c^{[\bar f]}(a)$. The Euler characteristic constraint (4.14b) is also satisfied since (4.10g) holds and the constraint itself is just multiplied by two. Since the right hand side of (4.10g) is s_e for $e = E(\{a,\mathrm{rev}(a)\})$ we have that $\hat s_e = \hat c^{[\bar f]}(a)$ and $\hat s_e = \hat c^{[\bar f]}(\mathrm{rev}(a))$ which shows (4.14c). Finally, $\hat s_a = \hat c^{[\bar f]}(a)$ also ensures $c^{[\bar f]}(a)\le 1$.

$\mathcal{P}^{+s}_{(4.14)} \subseteq \mathcal{P}_{(4.10)}$: Let $(\hat x,\hat c,\hat p)\in\mathcal{P}_{(4.14)}$ be arbitrary but fixed and $s_e := \hat c^{[\bar f]}(A(e))/2$ for all $e\in E$. The term $\hat c^{[\bar f]}(A)$ equals $2s(E)$ by the projection, thus the Euler characteristic constraint (4.10b) is satisfied. The projection together with (4.14c) ensure (4.10g). This also proves that (4.14e), (4.14f) and (4.14g) hold. □

4.6.2 Polynomially Sized Formulations and Speed-Ups

The size of the formulations presented so far is exponential in the graph size. We need $\sum_{v\in V} 2^{d_v}\in\mathcal{O}(2^{|V|})$ many CUT constraints to ensure that the rotations around the vertices consist of exactly one cycle. As presented in our previous work [Bey+16] a way to reduce the size of the formulation is to switch to alternative representations of rotation systems. We consider two approaches using betweenness variables and index variables. Both yield polynomially sized formulations.

Index Reformulation. We introduce binary variables $q^v_{j,u}$ that are 1 if and only if u is the j-th vertex in the rotation at v. It is easy to ensure that the q^v-variables form a bijective mapping

$[d_v] \to N(v)$ by

$$\sum\nolimits_{i\in[d_v]} q^v_{j,u} = 1 \qquad \forall v \in V,\ u \in N(v), \tag{4.16a}$$

$$\sum\nolimits_{u\in N(v)} q^v_{j,u} = 1 \qquad \forall v \in V,\ j \in [d_v]. \tag{4.16b}$$

There are two possibilities to integrate the new variables in existing formulations using p-variables:

- Connect the p- and q-variables by constraints that ensure the p-variables form a rotation system on subgraph induced by the edges $\{e : s_e = 1\}$ in the solution. Then the remaining constraints of formulation (4.10) can be re-used.
- Change the constraints (4.16) such that the q-variables only form a rotation on the solution graph $\{e : s_e = 1\}$. Then we can completely omit the p-variables and connect the q-variables to the containment variables by

$$c^i_{vw} \geq c^i_{uv} + (q^v_{j,u} + q^v_{j+1,w} - 2) \qquad \forall i \in [\bar{f}],\ v \in V,\ j \in [d_v],\ u, w \in N(v),$$
$$c^i_{uv} \geq c^i_{vw} + (q^v_{j,u} + q^v_{j+1,w} - 2) \qquad \forall i \in [\bar{f}],\ v \in V,\ j \in [d_v],\ u, w \in N(v).$$

Due to our experience with the index reformulation for MGP [Bey+16] we refrain from implementing this exact algorithm.

Betweenness Reformulation. As in Section 4.4 we use betweenness variables to describe the rotation system of the input graph G. We use variables $b^v_{x,y,z}$ for each triple of pairwise different $x, y, z \in N(v)$. By $b^v_{x,y,z} = 1$ we denote that y is somewhere between x and z in the counter-clockwise rotation at v.

Note that we can only use betweenness variables to represent the rotations of vertices with minimum degree three. We therefore split $V = V_2 \uplus V_+ := \{v \in V : d_v \leq 2\} \uplus \{c \in V : d_v \geq 3\}$. We delete all p^v-variables for vertices $v \in V_+$ and only keep them V_2.

The formulation using betweenness variables is:

$$\begin{aligned}
\max\quad & \sum\nolimits_{e\in E} w_e s_e && && 4.17\\
\text{s.t.}\quad & \text{(4.10b)–(4.10h)} && \text{constraints independent of } p^v &\\
& \text{(4.10i)–(4.10n)} && \text{but change “}V\text{” to “}V_2\text{”} &\\
& \text{(4.8b)–(4.8f)} && b\text{'s form a rot.-sys.} &\\
& c^i_{vw} \geq c^i_{uv} + \sum\nolimits_{x\in N(v)\setminus\{u,w\}} b^v_{u,w,x} - (d_v - 2) && \forall i \in [\bar{f}],\ v \in V_+,\ u, w \in N(v) & \text{(4.17a)}\\
& c^i_{uv} \geq c^i_{vw} + \sum\nolimits_{x\in N(v)\setminus\{u,w\}} b^v_{u,w,x} - (d_v - 2) && \forall i \in [\bar{f}],\ v \in V_+,\ u, w \in N(v) & \text{(4.17b)}\\
& s_e, x_i, c^i_a, p^v_{u,w} \in \{0,1\} && \text{see (4.10)},\ v \in V_2 &\\
& b^v_{x,y,z} \in \{0,1\} && \forall v \in V_+, \text{ p.d. } x, y, z \in N(v) &
\end{aligned}$$

Theorem. *Formulation* (4.17) *is polynomially sized and solves MPS.*

Proof. The size of the formulation (4.17) is polynomially bounded as we keep (4.10m) only for vertices v with $d_v \leq 2$.

The constraints (4.10b)–(4.10h) are shared with formulation (4.10). Furthermore, we have a rotation system on all vertices in V_2 and the according p^v-variables are connected to the

c-variables as before by (4.10i)–(4.10n). The borrowed constraints (4.8b)–(4.8f) ensure that we have a rotation system on V_+ and thus a rotation system for the whole graph. The connection of the betweenness variables with the containment variables is done in the two new constraints:

If w is the direct successor of u in the rotation around v (*i.e.*, $p^v_{u,w} = 1$ in formulation (4.17)) we have

$$\sum_{x \in N(v) \setminus \{u,w\}} b^v_{u,w,x} = d_v - 2.$$

Thus, the constraints (4.17a) and (4.17b) simulate the face traversal procedure. □

Again, due to our experience with the betweenness reformulation for MGP [Bey+16] we refrain from implementing this exact algorithm.

Degree Three Vertices. The models presented in the last two paragraphs are reformulations to achieve polynomially sized formulations. As in our work on MGP algorithms we suppose that there are several methods to speed-up the algorithms in practice, where the exponential size is not a problem.

Thus, we take care of the special structure of the neighborhood of degree three vertices. We redefine $V_3 := \{v \in V : d_v = 3\}$ and $V_+ := V \setminus V_3$. We use the same formulation (4.10a)–(4.10n) as above, but we exchange (4.10i)–(4.10n) to work only on V_3:

4.10″

$$c^i_{vw} \geq c^i_{uv} + p^v_{u,w} - 1 \qquad \forall i \in [\bar{f}],\, v \in V_+,\, u, w \in N(v) \qquad (4.10\text{i}'')$$

$$c^i_{uv} \geq c^i_{vw} + p^v_{u,w} - 1 \qquad \forall i \in [\bar{f}],\, v \in V_+,\, u, w \in N(v) \qquad (4.10\text{j}'')$$

$$p^v(u \times N(v)) = s_{vu} \qquad \forall v \in V_+,\, vu \in A \qquad (4.10\text{k}'')$$

$$p^v(N(v) \times w) = s_{vw} \qquad \forall v \in V_+,\, vw \in A \qquad (4.10\text{l}'')$$

$$p^v(U \times \bar{U}) \geq s_v(\{u, \tilde{u}\}) - 1 \qquad \forall v \in V_+,\, \forall \emptyset \neq U \subsetneq N(v),\, \forall u \in U,\, \forall \tilde{u} \in \bar{U} \qquad (4.10\text{m}'')$$

$$p^v_{u,u} + s_{vw} \leq 1 \qquad \forall v \in V_+,\, \forall u \neq w \in N(v) \qquad (4.10\text{n}'')$$

For a degree three vertex v the variable p^v describes if its neighborhood $N(v) =: \{n^v_0, n^v_1, n^v_2\}$ appears as $n^v_0 \to n^v_1 \to n^v_2 \to n^v_0$ (when $p^v = 1$) or $n^v_0 \to n^v_2 \to n^v_1 \to n^v_0$ (when $p^v = 0$). Note that we have an important difference in the usage of the old $p^v_{u,w}$ variables compared to the new p^v variables for degree three nodes: If $p^v_{u,w} = 1$ we have $u \to w$ in the cyclic order of $N(v)$ and both arcs $u \to v$ and $v \to w$ have to be in the solution. The new p^v variables describe a rotation of $N(v)$ independent of if all neighbors are adjacent to v in the solution subgraph.

The equivalents of (4.10i″) and (4.10j″)—which are only defined on V_+—on V_3 are:

4.18

$$c^i_{vn^v_{j+1}} \geq c^i_{n^v_j v} + (p^v - 1) + (s_{vn^v_{j+1}} - 1) \qquad \forall i \in [\bar{f}],\, v \in V_3,\, j \in [3] \qquad (4.18a)$$

$$c^i_{n^v_j v} \geq c^i_{vn^v_{j+1}} + (p^v - 1) + (s_{n^v_j v} - 1) \qquad \forall i \in [\bar{f}],\, v \in V_3,\, j \in [3] \qquad (4.18b)$$

$$c^i_{vn^v_j} \geq c^i_{n^v_{j+1} v} - p^v + (s_{vn^v_j} - 1) \qquad \forall i \in [\bar{f}],\, v \in V_3,\, j \in [3] \qquad (4.18c)$$

$$c^i_{n^v_{j+1} v} \geq c^i_{vn^v_j} - p^v + (s_{n^v_{j+1} v} - 1) \qquad \forall i \in [\bar{f}],\, v \in V_3,\, j \in [3] \qquad (4.18d)$$

$$c^i_{vn^v_{j+2}} \geq c^i_{n^v_j v} + (p^v - 1) + (s_{vn^v_{j+2}} - 1) - s_{vn^v_{j+1}} \qquad \forall i \in [\bar{f}],\, v \in V_3,\, j \in [3] \qquad (4.18e)$$

$$c^i_{n^v_j v} \geq c^i_{vn^v_{j+2}} + (p^v - 1) + (s_{n^v_j v} - 1) - s_{vn^v_{j+1}} \qquad \forall i \in [\bar{f}],\, v \in V_3,\, j \in [3] \qquad (4.18f)$$

$$c^i_{vn^v_j} \geq c^i_{n^v_{j+2} v} - p^v + (s_{vn^v_j} - 1) - s_{vn^v_{j+1}} \qquad \forall i \in [\bar{f}],\, v \in V_3,\, j \in [3] \qquad (4.18g)$$

$$c^i_{n^v_{j+2} v} \geq c^i_{vn^v_j} - p^v + (s_{vn^v_{j+2}} - 1) - s_{vn^v_{j+1}} \qquad \forall i \in [\bar{f}],\, v \in V_3,\, j \in [3] \qquad (4.18h)$$

The new constraints (4.18a)–(4.18d) describe the case where the degree of v in the solution is also three. Here (4.18a)–(4.18b) are for the $p^v = 1$ case, and (4.18c)–(4.18d) are for the $p^v = 0$

case. We use (4.18e)–(4.18h) in the same way when the degree of v in the solution is two. There are no constraints needed for a degree-1 vertex in the solution.

Note that we do not need any further constraints to ensure that the new p^v variables on V_3 form a rotation system.

Low Degree Vertices. The principle of the last paragraph can be extended to vertices v of arbitrary fixed degree $d_v \geq 4$. There are $\varrho := (d_v - 1)!$ different rotations. Instead of using $\mathcal{O}(d_v^2)$ many variables $p^v_{u,w}$, we introduce $\lceil \log_2 \varrho \rceil$ binary variables and represent the index of the rotation as a binary number. Since this process is coupled with a substantial trade-off of more complicated and weaker constraints, we refrain from using it for $d_v \geq 4$.

Other Speed-Ups, PBSs and Final Parameter Setting. In Chapter 2 on the Minimum Genus Problem we discussed more speed-up techniques. Some of them only apply for SAT/PBS-based formulations (such as incremental formulations using warm-starts of SAT/PBS-solvers) or are not necessary in ILP formulations. Remember the paragraph "Binary Face Representations" on page 30, where we discussed a problem ($f^2|E|$ constraints ($\mathbb{B}C_1^{\mathrm{I}}$) to prohibit that a dart appears in multiple facial walks) that only occurs in SAT formulations. The particular problem discussed in that paragraph can be expressed in a simple ILP constraint ($\mathbb{I}C_1^{\mathrm{I}}$).

Note that all formulations presented in this section can easily be transformed into SAT/PBS formulations. We refrain from explicitly providing the transformed versions as we already did that in Chapter 2 for the Minimum Genus Problem, where we worked with similar exact algorithms.

Based on the experimental results in [Bey+16], we use the initial formulation (4.10) (without the unnecessary constraints (4.10n)) with the speed-up method for degree-three vertices. Again, we use the greedy Cactus algorithm [Căl+98] (extended by a naive approach to make the solution also a maximal planar subgraph) as initial solution for our formulation.

4.7 Experimental Evaluation: Different Formulations and Overall Practicality

In this section we compare the developed exact algorithms

(1) Kuratowski-based ILP (ILP Kurat.) with threshold sequence $0.01, 0.1$, heap size 50 and one heuristic run after each new LP solution with randomization factor 0.5,

(2) Kuratowski-based PBS (PBS Kurat.) using warm-starts of the PBS solver to iteratively add new constraints for further Kuratowski subdivisions,

(3) Total Orders-based ILP (ILP Orders) without partial orders and without betweenness variables,

(4) Total Orders-based PBS (PBS Orders) with the same configuration in (3),

(5) Facial Walks-based ILP (ILP Faces) in the exponentially sized formulation with degree-three speed-up,

(6) Facial Walks-based PBS (PBS Faces) with the same configuration in (5),

and the implementation for

(7) c-planarity which is already part of the OGDF (OGDF c-planar).

We used the 8 249 non-planar *Rome* graphs, 423 non-planar *North* graphs, 580 non-planar *Expander* graphs and 105 non-planar *SteinLib* instances. See Section 1.4 for details on the instance sets.

Our C++ code is compiled with GCC 5.3.1-13, and runs on a single core of an Intel(R) Xeon(R) CPU E5-2430 v2 with DDR3 Memory @ 1600 MHz under Debian 8. We use the ILP solver SCIP 4.0 with Gurobi 7 as LP solver, the PBS solver Clasp 3.2.1, the OGDF, and apply a memory limit of 8 GB as well as a time limit of 20 minutes.

MPS Computation Framework. As for the Minimum Genus Problem (see [Bey+16, Section 3] or Section 2.6 on page 34), we compute the MPS of each biconnected component B_i separately. For each B_i, we compute the non-planar core (C_i, w_i) which yields an instance for the Weighted Maximum Planar Subgraph Problem with graph C_i and weights w_i. See Lemma 1.4.

Results on the Real World Graphs Rome and North. We use the PBS and ILP approaches to compute the MPS of the *Rome* and *North* instances mentioned above. A computation is called *successful* if it finished within 20 minutes and used not more than 8 GB memory.

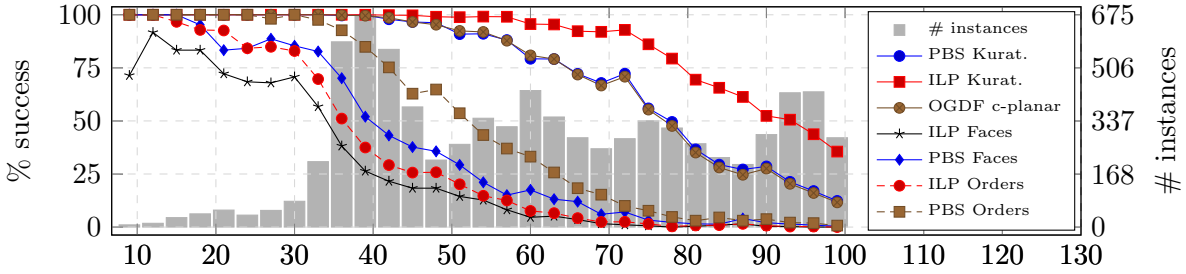
Figures 4.F and 4.E show the success rates for the *North* and *Rome* instances, respectively. In both subfigures (a) we compare the success rates grouped by the number of nodes of the input graph. For the *North* and *Rome* graphs, a group is defined by clustering the number of nodes to the nearest multiple of 5 and 3, respectively. In both subfigures (b) the groups are defined by clustering the number of edges to the nearest multiple of 10 and 5 for *North* and *Rome*, respectively. The groups in the remaining subfigures are defined analogously but for the number of nodes and edges of the non-planar core.

For the *Rome* graphs (Figure 4.E) the success rates of the different approaches are can be clearly distinguished from each other. ILP Kurat. has the best performance and is able to solve more than 90 % of the instances on up to 70 nodes. Surprisingly, the results of its PBS variant are almost equal to the performance of OGDF c-planar. Instances with up to 55 nodes can be solved with a high success rate. For both criteria, Total Orders and Facial Walks, the PBS variant is always superior to the ILP formulation. When we compare the performance against the number of edges of the input graphs, we get similar results. Here, ILP Kurat. is able to solve instances on up to 105 edges with a high success rate. When looking at the plots, clustered by NPC size, the picture becomes clearer.

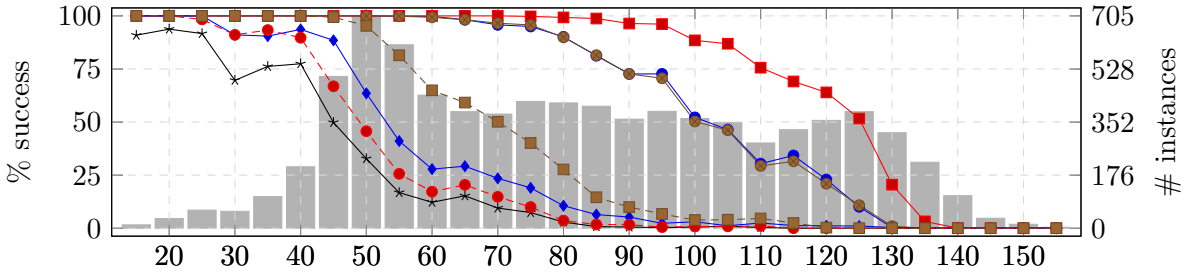
The results for the *North* graphs are not as unambiguous as for the *Rome* graphs. ILP Kurat. still has the best performance and is able to solve instances on up to 45 nodes and 80 edges with a high success rate. However, on the *North* instances, it no longer holds that the PBS variants of Total Orders and Facial Walks are always as good or better than their ILP counterparts.

In Figure 4.G, we consider the success rates grouped by the density of the graph ($|E|/|V|$) and the NPC ($|E_{\text{NPC}}|/|V_{\text{NPC}}|$), respectively. On the *North* instances, the performance for graphs with NPC density above 2.4 drops significantly. As the NPCs of the *Rome* graphs are sparse, we do not discover this pattern on that instance set. We see the same characteristics as in the figures before.

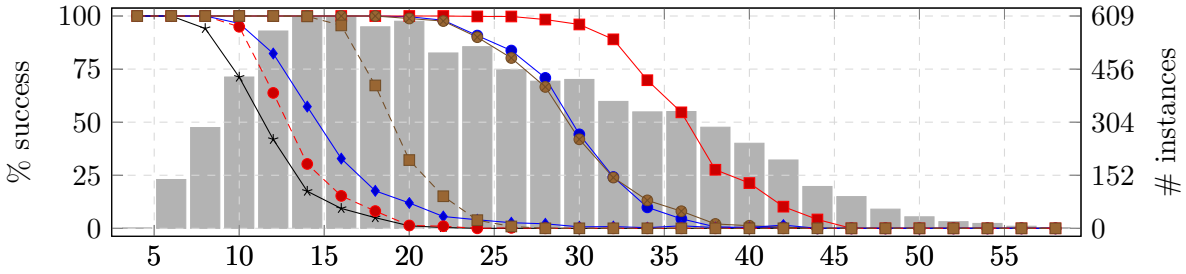
In Figure 4.H, we compare the average run-times on the set of instances solved by all algorithms, we call it the *common* set. Note the differences between, *e.g.*, Figure 4.E(a) and 4.H(b): The common set is significantly smaller than the set of all instances. This results in surprisingly small run-times. Most of the algorithms with a low success rate found an optimal solution in a small amount of time or did not find a solution at all. Based on this behavior, we only focused on the variant with the highest success rate for each criterion in the subfigures (c) and (d). Here, the sizes of the common set with respect to these four algorithms allow more accurate statements. We see that ILP Kurat. almost always dominates the other approaches not



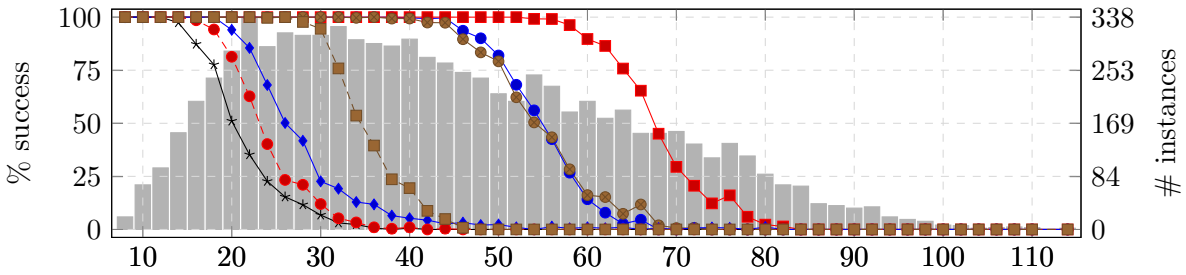
(a) Rome: success rate per input graph nodes



(b) Rome: success rate per input graph edges

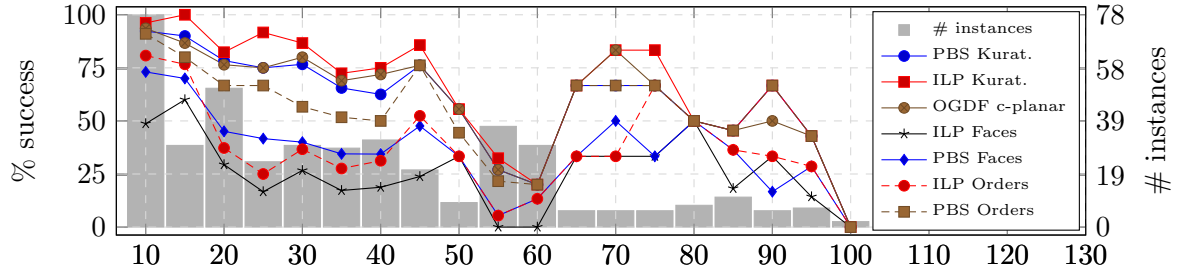


(c) Rome: success rate per NPC nodes

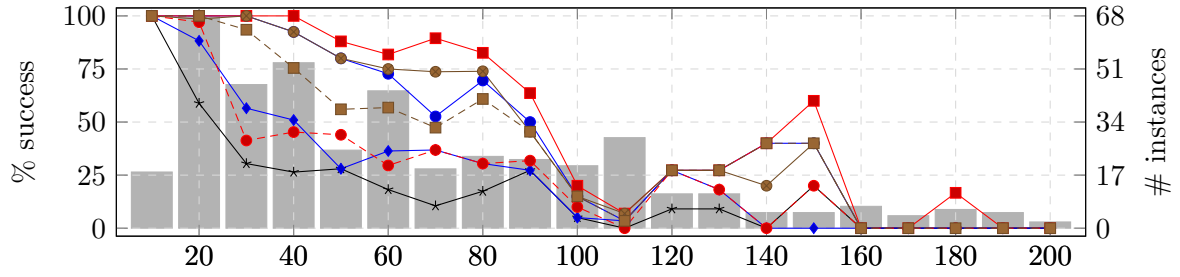


(d) Rome: success rate per NPC edges

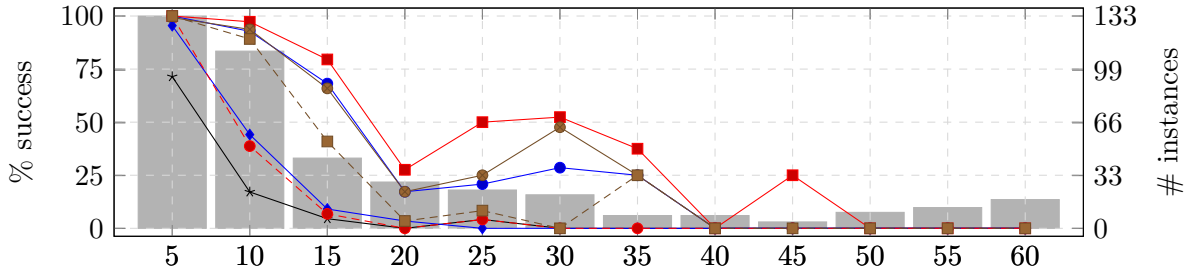
Figure 4.E: Success rates on the *Rome* instances. We compare against the number of nodes, edges, NPC nodes and NPC edges. The number of nodes in (a) is clustered to the nearest multiple of 3. We use the same clustering with different sizes in the other subfigures: (b) 5, (c) 2, (d) 2. The gray bars show the number of instances each cluster. The legend of (a) applies to all other subfigures. 4.E



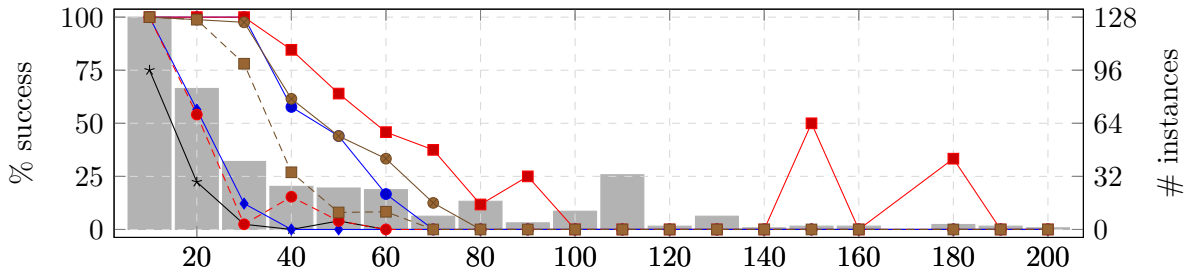
(a) North: success rate per input graph nodes



(b) North: success rate per input graph edges



(c) North: success rate per NPC nodes



(d) North: success rate per NPC edges

4.F **Figure 4.F:** Success rates on the *North* instances. We compare against the number of nodes, edges, NPC nodes and NPC edges. The number of nodes in (a) is clustered to the nearest multiple of 5. We use the same clustering with different sizes in the other subfigures: (b) 10, (c) 5, (d) 10. The gray bars show the number of instances each cluster. The legend of (a) applies to all other subfigures.

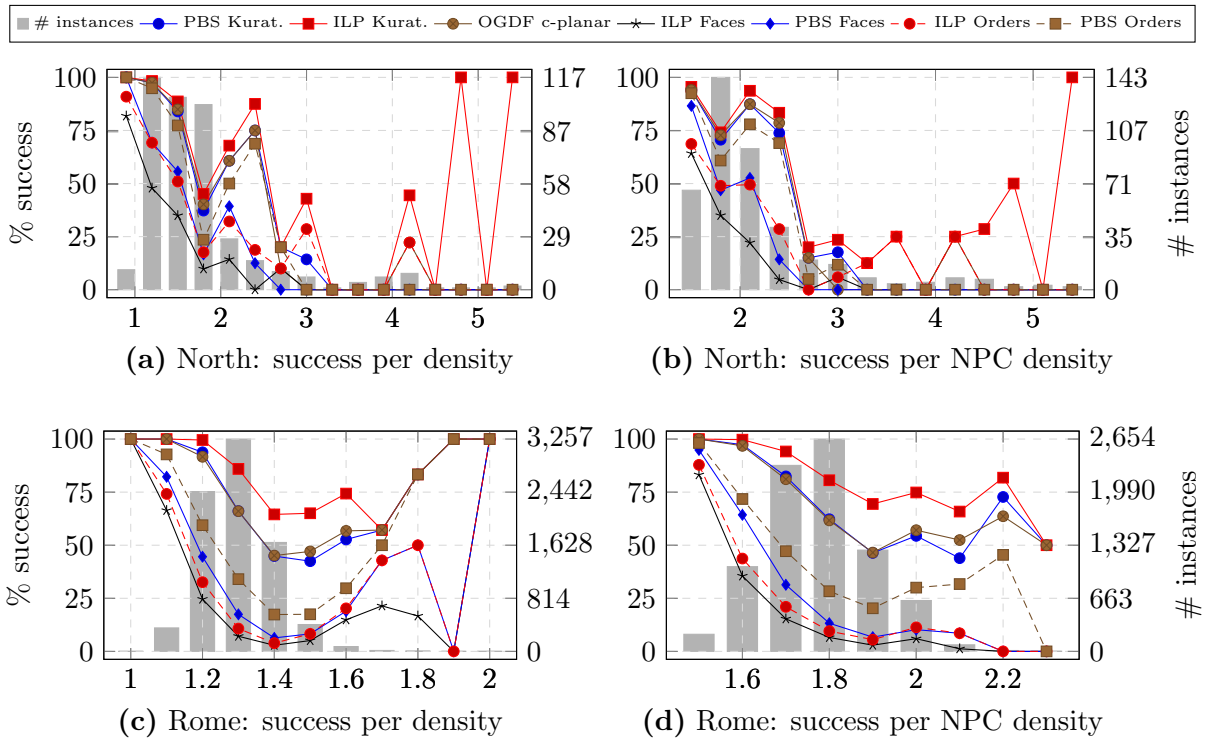


Figure 4.G: Success rates on the *Rome* and *North* instances compared against the density and NPC density. The results of (a) and (b) are clustered in intervals of length 0.3. The results of (c) and (d) are clustered in intervals of length 0.1 4.G

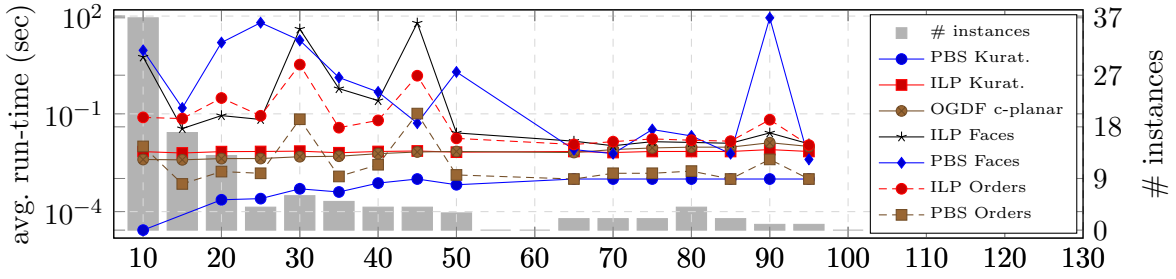
only in the success rate but also from the run-time point of view. The run-times of the Facial Walks and Total Orders approaches differ by orders of magnitude compared to ILP Kurat.

Results on the SteinLib and Expander Instances. Tables 4.I and 4.K show the number of solved instances of the *Expander* and *SteinLib* instances, respectively.

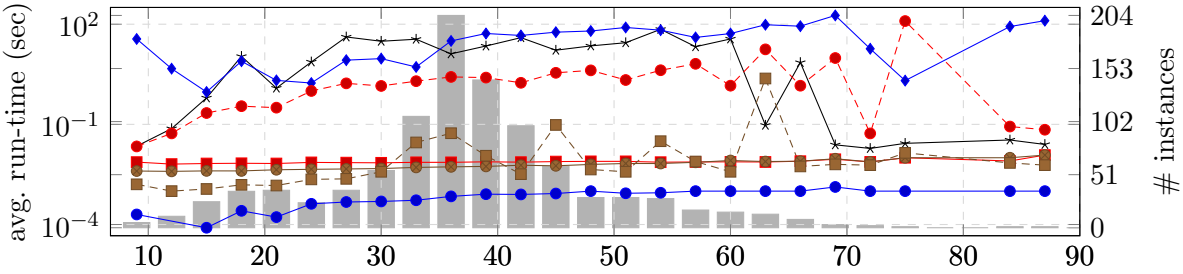
We were able to solve almost all *Expander* instances on 10 vertices with almost all approaches. Only the performance of the ILP variant for the Facial Walk criterion and PBS Faces on the configuration ($|V| = 10, d_v = 6$) are significantly worse compared to the other approaches. Our ILP Kurat. algorithm is able to solve many of the instances on up to 50 nodes. However, we have the case that we can solve all 20 instances of the configurations ($|V| = 20, d_v = 4$) and ($|V| = 20, d_v = 10$) but none of the instances with configuration ($|V| = 20, d_v = 6$). The same behavior occurs for other configurations. The Total Orders and Facial Walk algorithms were not able to solve an instance on 30 or more nodes.

The one instance on 10 000 nodes that was solved by OGDF's exact algorithm for c-planarity is a non-planar graph where OGDF's algorithm claims one has to delete zero edges to obtain a MPS. This obviously points out an error in this implementation.

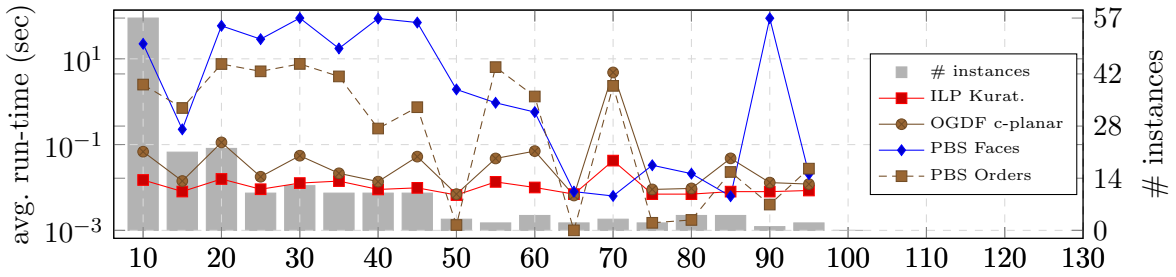
The run-times compared in Table 4.J show that we reach the limit of our algorithms already for $|V| \leq 20$ in this instance class. Also the exponential dependence of the run-time on the input size becomes visible even on the solved instances. The reason for the surprisingly low run-time of ILP Orders in the configuration ($|V| = 10, d_v = 6$) is that it solved one instance more than its PBS variant and it has a disproportionate high run-time only on this single instance. We omit the results for ILP Faces in this table as the common set including this algorithm would become to small for accurate statements.



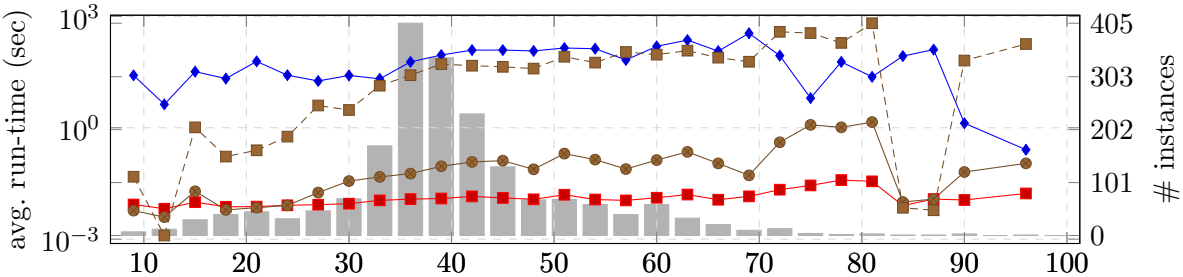
(a) North: average run-time per number of nodes on the common solved set



(b) Rome: average run-time per number of nodes on the common solved set



(c) North: average run-time per number of nodes on the common solved set



(d) Rome: average run-time per number of nodes on the common solved set

4.H **Figure 4.H:** Average run-time (in seconds) on the *North* and *Rome* instances. We cluster as in Figure 4.F and 4.E. The legends of (a) and (c) apply to (b) and (d), respectively. In (a) and (b) the common set is defined with respect to all algorithms. In (c) and (d) the common set is defined only by the variant with highest success rate for each criterion.

solved instances		OGDF	Kurat.		Orders		Faces	
$\|V\|$	d_v	c-planar	ILP	PBS	ILP	PBS	ILP	PBS
10	4	20	20	20	18	20	3	20
10	6	20	20	20	18	17	2	1
20	4	20	20	18	0	0	0	1
20	10	0	20	0	0	0	0	0
30	4	0	12	0	0	0	0	0
30	20	0	20	0	0	0	0	0
50	40	0	20	0	0	0	0	0
10 000	20	1	0	0	0	0	0	0

Table 4.I: Number of solved instances for *Expanders*. For configurations $(|V|, d_v)$ that are omitted, no algorithm solved any instance. The instance set contains 20 graphs per configuration. 4.I

avg. run-time		OGDF	Kurat.		Orders		Faces	
$\|V\|$	d_v	c-planar	ILP	PBS	ILP	PBS	ILP	PBS
10	4	0.0108	0.0094	0.0010	196.2551	0.5858	—	124.9892
10	6	0.9934	0.0295	4.8399	28.0698	193.0456	—	—
20	4	15.5954	1.0990	18.4400	—	—	—	—

Table 4.J: Average run-time (in seconds) for the *Expander* graphs on the set of instances solved by all algorithms (except for the variants marked with "—"). 4.J

From the 580 tested *SteinLib* instance we were only able to solve 10 instances via ILP Kurat. Again, ILP Kurat. dominates all other approaches. No instance of the i080 class was solved.

4.8 Summary and Conclusion

There is a variety of criteria for planarity. We developed and implemented ILP- and PBS-based exact algorithms based on three criteria: Kuratowski subdivisions, simulating facial walks and total orders. The latter two allow polynomially sized formulations.

However, our experiments on *Rome*, *North*, *Expander* and *SteinLib* graphs showed that the ILP-based exact algorithm using Kuratowski subdivisions dominates all other approaches both in success rate and run-time. Hence, we were not able to develop a new algorithm that beats the performance of Kuratowski based formulations.

In some of the formulations we used the experience gained in our work [Bey+16] on the Minimum Genus Problem to set parameters of our algorithms. It remains an open task to optimize the parameter settings in such cases.

solved instances		OGDF	Kurat.		Orders		Faces	
instance class	# instances	c-planar	ILP	PBS	ILP	PBS	ILP	PBS
B	18	7	9	8	3	3	2	1
single	7	1	1	1	0	1	0	0

Table 4.K: Number of solved instances for the *SteinLib* instance class. No instance of i080 was solved. 4.K

Still, we did not evaluate all criteria that may be used for practical exact algorithms and hope that newly derived formulations or entirely new formulations based on other criteria are able to perform better than the standard approach using Kuratowski subdivisions.

Bibliography

[ABL95] Dan Archdeacon, C. Paul Bonnington, and Charles H. C. Little. "An algebraic characterization of planar graphs." In: *Journal of Graph Theory* 19.2 (1995), pp. 237–250. ISSN: 0364-9024. DOI: 10.1002/jgt.3190190209.

[ALS91] Stefan Arnborg, Jens Lagergren, and Detlef Seese. "Easy Problems for Tree-Decomposable Graphs." In: *Journal of Algorithms. Cognition, Informatics and Logic* 12.2 (1991), pp. 308–340. ISSN: 0196-6774. DOI: 10.1016/0196-6774(91)90006-K.

[Arc86] Dan Archdeacon. "The Orientable Genus Is Nonadditive." In: *Journal of Graph Theory* 10.3 (1986), pp. 385–401. ISSN: 0364-9024. DOI: 10.1002/jgt.3190100314.

[AŠ98] Dan Archdeacon and Josef Širáň. "Characterizing Planarity Using Theta Graphs." In: *Journal of Graph Theory* 27.1 (1998), pp. 17–20. ISSN: 0364-9024. DOI: 10.1002/(SICI)1097-0118(199801)27:1<17::AID-JGT4>3.0.CO;2-J.

[Bat+62] Joseph Battle, Frank Harary, Yukihiro Kodama, and John William Theodore Youngs. "Additivity of the genus of a graph." In: *Bulletin of the American Mathematical Society* 68 (1962), pp. 565–568. ISSN: 0002-9904. DOI: 10.1090/S0002-9904-1962-10847-7.

[BD09] Henning Bruhn and Reinhard Diestel. "MacLane's theorem for arbitrary surfaces." In: *Journal of Combinatorial Theory. Series B* 99.2 (2009), pp. 275–286. ISSN: 0095-8956. DOI: 10.1016/j.jctb.2008.03.005.

[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: *Experimental Algorithms - 15th International Symposium, SEA 2016, St. Petersburg, Russia, June 5-8, 2016, Proceedings*. Ed. by Andrew V. Goldberg and Alexander S. Kulikov. Vol. 9685. Lecture Notes in Computer Science. Springer, 2016, pp. 75–88. ISBN: 978-3-319-38850-2. DOI: 10.1007/978-3-319-38851-9_6.

[Bie14] Armin Biere. "Yet another Local Search Solver and Lingeling and Friends Entering the SAT Competition 2014." In: *Proceedings of SAT Competition 2014: Solver and Benchmark Descriptions*. Ed. by Anton Belov, Daniel Diepold, Marijn J.H. Heule, and Matti Järvisalo. Series of Publications B B-2014-2. University of Helsinki, Dept. Comp. Sc. 2014, pp. 39–40. ISBN: 978-951-51-0043-6.

[BM04] 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* 8.3 (2004), pp. 241–273. ISSN: 1526-1719. DOI: 10.7155/jgaa.00091.

[Bod86] Hans L. Bodlaender. *Classes of graphs with bounded tree-width*. Tech. rep. RUU-CS-86-22. Department of Information and Computing Sciences, Utrecht University, 1986. URL: http://www.cs.uu.nl/research/techreps/RUU-CS-86-22.html.

[BR06] Joseph P. Bohanon and Les Reid. "Finite groups with planar subgroup lattices." In: *Journal of Algebraic Combinatorics. An International Journal* 23.3 (2006), pp. 207–223. ISSN: 0925-9899. DOI: 10.1007/s10801-006-7392-8.

[BS88] Matthew G. Brin and Craig C. Squier. "On the Genus of $Z_3 \times Z_3 \times Z_3$." In: *European Journal of Combinatorics* 9.5 (1988), pp. 431–443. ISSN: 0195-6698. DOI: 10.1016/S0195-6698(88)80002-7.

[Buc+05] Christoph Buchheim, Dietmar Ebner, Michael Jünger, Gunnar W. Klau, Petra Mutzel, and René Weiskircher. "Exact Crossing Minimization." In: *Graph Drawing, 13th International Symposium, GD 2005, Limerick, Ireland, September 12-14, 2005, Revised Papers*. Ed. by Patrick Healy and Nikola S. Nikolov. Vol. 3843. Lecture Notes in Computer Science. Springer, 2005, pp. 37–48. ISBN: 3-540-31425-3. DOI: 10.1007/11618058_4.

[Căl+98] Gruia Călinescu, Cristina Gomes Fernandes, Ulrich Finkler, and Howard Karloff. "A Better Approximation Algorithm for Finding Planar Subgraphs." In: *Journal of Algorithms. Cognition, Informatics and Logic* 27.2 (1998). 7th Annual ACM-SIAM Symposium on Discrete Algorithms (Atlanta, GA, 1996), pp. 269–302. ISSN: 0196-6774. 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: *Math. Program. Comput.* 3.3 (2011), pp. 261–280. DOI: 10.1007/s12532-011-0027-7.

[Cay78] Arthur Cayley. "Desiderata and Suggestions: No. 2. The Theory of Groups: Graphical Representation." In: *American Journal of Mathematics* 1.2 (1878), pp. 174–176. ISSN: 0002-9327. DOI: 10.2307/2369306.

[CCE13] Sergio Cabello, Erin W. Chambers, and Jeff Erickson. "Multiple-source shortest paths in embedded graphs." In: *SIAM Journal on Computing* 42.4 (2013), pp. 1542–1571. ISSN: 0097-5397. DOI: 10.1137/120864271.

[Čer80] A. A. Černjak. "On Little's conjecture on planar graphs." In: *Vescī Akadèmīī Navuk BSSR. Seryja Fīzīka-Matèmatyčnyh Navuk* 2 (1980), pp. 41–45, 140. ISSN: 0002-3574.

[CG09] Markus Chimani and Carsten Gutwenger. "Non-planar core reduction of graphs." In: *Discrete Mathematics* 309.7 (2009), pp. 1838–1855. ISSN: 0012-365X. DOI: 10.1016/j.disc.2007.12.078.

[CG15] Marston Conder and Ricardo Grande. "On embeddings of circulant graphs." In: *Electronic Journal of Combinatorics* 22.2 (2015), Paper 2.28, 27. ISSN: 1077-8926.

[Cha+15] Timothy M. Chan, Fabrizio Frati, Carsten Gutwenger, Anna Lubiw, Petra Mutzel, and Marcus Schaefer. "Drawing Partially Embedded and Simultaneously Planar Graphs." In: *Journal of Graph Algorithms and Applications* 19.2 (2015), pp. 681–706. ISSN: 1526-1719. DOI: 10.7155/jgaa.00375.

[Cha02] John Chambers. "Hunting for Torus Obstructions." M.Sc. thesis. University of Victoria, 2002.

[Chi+09] Markus Chimani, Maria Kandyba, Ivana Ljubic, and Petra Mutzel. "Obtaining optimal k-cardinality trees fast." In: *ACM Journal of Experimental Algorithmics* 14.5 (2009). DOI: 10.1145/1498698.1537600.

[Chi+13] Markus Chimani, Carsten Gutwenger, Michael Jünger, 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: crcpress.com/Handbook-of-Graph-Drawing-and-Visualization/Tamassia/9781584884125.

[Chi08] Markus Chimani. "Computing Crossing Numbers." PhD thesis. Department of Computer Sciene, Technical University Dortmund, 2008. URL: http://tcs.uos.de/_media/pubs/computingcrossingnumbers_phdthesis_chimani_pdf.pdf.

[Chi09] Markus Chimani. "Computing Crossing Numbers." In: *Ausgezeichnete Informatikdissertationen 2008*. Ed. by Dorothea Wagner. Vol. D-9. LNI. GI, 2009, pp. 41–50. ISBN: 978-3-88579-413-4.

[CHW16] Markus Chimani, Ivo Hedtke, and Tilo Wiedera. "Limits of Greedy Approximation Algorithms for the Maximum Planar Subgraph Problem." In: *Combinatorial Algorithms - 27th International Workshop, IWOCA 2016, Helsinki, Finland, August 17-19, 2016, Proceedings*. Ed. by Veli Mäkinen, Simon J. Puglisi, and Leena Salmela. Vol. 9843. Lecture Notes in Computer Science. Springer, 2016, pp. 334–346. ISBN: 978-3-319-44542-7. DOI: 10.1007/978-3-319-44543-4_26.

[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)*. Ed. by Yifan Hu and Martin Nöllenburg. Vol. abs/1609.02443. CoRR, 2016.

[CL91] Jason (Jingsheng) Cong and C. L. Liu. "On the k-Layer Planar Subset and Topological Via Minimization Problems." In: *IEEE Trans. on CAD of Integrated Circuits and Systems* 10.8 (1991), pp. 972–981. DOI: 10.1109/43.85735.

[CMB08] Markus Chimani, Petra Mutzel, and Immanuel M. Bomze. "A New Approach to Exact Crossing Minimization." In: *Algorithms - ESA 2008, 16th Annual European Symposium, Karlsruhe, Germany, September 15-17, 2008. Proceedings*. Ed. by Dan Halperin and Kurt Mehlhorn. Vol. 5193. Lecture Notes in Computer Science. Springer, 2008, pp. 284–296. ISBN: 978-3-540-87743-1. 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: *Graph Drawing, 15th International Symposium, GD 2007, Sydney, Australia, September 24-26, 2007. Revised Papers*. Ed. by Seok-Hee Hong, Takao Nishizeki, and Wu Quan. Vol. 4875. Lecture Notes in Computer Science. Springer, 2007, pp. 159–170. ISBN: 978-3-540-77536-2. DOI: 10.1007/978-3-540-77537-9_17.

[Col93] Yves Colin de Verdière. "On a new graph invariant and a criterion for planarity." In: *Graph structure theory (Seattle, WA, 1991)*. Vol. 147. Contemp. Math. Amer. Math. Soc., Providence, RI, 1993, pp. 137–147. DOI: 10.1090/conm/147/01168.

[Coo+98] William J. Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver. *Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization. 1998, pp. x+355. ISBN: 0-471-55894-X.

[CR16] Ricky X. F. Chen and Christian M. Reidys. "On the local genus distribution of graph embeddings." In: *ArXiv e-prints* (Jan. 2016). eprint: 1601.02574.

[CR17] Ricky X. F. Chen and Christian M. Reidys. "On the local genus distribution of graph embeddings." In: *Journal of Combinatorial Mathematics and Combinatorial Computing* 101 (May 2017), pp. 157–173.

[CS13] Chandra Chekuri and Anastasios Sidiropoulos. "Approximation Algorithms for Euler Genus and Related Problems." In: *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*. IEEE Computer Society, 2013, pp. 167–176. ISBN: 978-0-7695-5135-7. DOI: 10.1109/FOCS.2013.26.

[CS17] Parinya Chalermsook and Andreas Schmid. "Finding Triangles for Maximum Planar Subgraphs." In: *WALCOM: Algorithms and Computation, 11th International Conference and Workshops, WALCOM 2017, Hsinchu, Taiwan, March 29-31, 2017, Proceedings.* Ed. by Sheung-Hung Poon, Md. Saidur Rahman, and Hsu-Chun Yen. Vol. 10167. Lecture Notes in Computer Science. Springer, 2017, pp. 373–384. ISBN: 978-3-319-53924-9. DOI: 10.1007/978-3-319-53925-6_29.

[CZ15] Markus Chimani and Robert Zeranski. "Upward Planarity Testing in Practice: SAT Formulations and Comparative Study." In: *ACM Journal of Experimental Algorithmics* 20 (2015), Article 1.2, 27. DOI: 10.1145/2699875.

[Dez+00] Michel Deza, Patrick W. Fowler, A. Rassat, and Kevin M. Rogers. "Fullerenes as Tilings of Surfaces." In: *Journal of Chemical Information and Computer Sciences* 40.3 (2000), pp. 550–558. DOI: 10.1021/ci990066h.

[DFF85] M. E. Dyer, L. R. Foulds, and A. M. Frieze. "Analysis of heuristics for finding a maximum weight planar subgraph." In: *European Journal of Operational Research* 20.1 (1985), pp. 102–114. ISSN: 0377-2217. DOI: 10.1016/0377-2217(85)90288-7.

[Di +00] Giuseppe Di Battista, Ashim Garg, Giuseppe Liotta, Roberto Tamassia, Emanuele Tassinari, Francesco Vargiu, and Luca Vismara. "Drawing Directed Acyclic Graphs: An Experimental Study." In: *International Journal of Computational Geometry & Applications* 10.6 (2000), pp. 623–648. DOI: 10.1142/S0218195900000358.

[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.

[DM41] Ben Dushnik and E. W. Miller. "Partially ordered sets." In: *American Journal of Mathematics* 63 (1941), pp. 600–610. ISSN: 0002-9327. DOI: 10.2307/2371374.

[DP15] Kosta Došen and Zoran Petrić. "A planarity criterion for graphs." In: *SIAM Journal on Discrete Mathematics* 29.4 (2015), pp. 2160–2165. ISSN: 0895-4801. DOI: 10.1137/140954957.

[DR91] Hristo Djidjev and John H. Reif. "An Efficient Algorithm for the Genus Problem with Explicit Construction of Forbidden Subgraphs." In: *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*. Ed. by Cris Koutsougeras and Jeffrey Scott Vitter. ACM, 1991, pp. 337–347. ISBN: 0-89791-397-3. DOI: 10.1145/103418.103456.

[Edm60] J. Edmonds. "A combinatorial representation for polyhedral surfaces." In: *Notices of the American Mathematical Society* 7 (1960), p. 646.

[EFN12] Jeff Erickson, Kyle Fox, and Amir Nayyeri. "Global minimum cuts in surface embedded graphs." In: *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*. ACM, New York, 2012, pp. 1309–1318. DOI: 10.1137/1.9781611973099.103.

[FC07] Cristina Gomes Fernandes and Gruia Călinescu. "Maximum Planar Subgraph." In: *Handbook of Approximation Algorithms and Metaheuristics*. Ed. by Teofilo F. Gonzalez. Chapman and Hall/CRC, 2007. ISBN: 978-1-58488-550-4. DOI: 10.1201/9781420010749.ch56.

[FCE95] Qing-Wen Feng, Robert F. Cohen, and Peter Eades. "Planarity for Clustered Graphs." In: *Algorithms - ESA '95, Third Annual European Symposium, Corfu, Greece, September 25-27, 1995, Proceedings*. Ed. by Paul G. Spirakis. Vol. 979. Lecture Notes in Computer Science. Springer, 1995, pp. 213–226. ISBN: 3-540-60313-1. DOI: 10.1007/3-540-60313-1_145.

[FF56] Lester Randolph Ford Jr. and Delbert Ray Fulkerson. "Maximal flow through a network." In: *Canadian Journal of Mathematics. Journal Canadien de Mathématiques* 8 (1956), pp. 399–404. ISSN: 0008-414X. DOI: 10.4153/CJM-1956-045-5.

[Fil78] I. S. Filotti. "An efficient algorithm for determining whether a cubic graph is toroidal." In: *Conference Record of the Tenth Annual ACM Symposium on Theory of Computing (San Diego, Calif., 1978)*. ACM, New York, 1978, pp. 133–142.

[Fis+94] Matteo Fischetti, Horst W. Hamacher, Kurt Jørnsten, and Francesco Maffioli. "Weighted k-cardinality trees: Complexity and polyhedral structure." In: *Networks* 24.1 (1994), pp. 11–21. DOI: 10.1002/net.3230240103.

[FMR79] I. S. Filotti, Gary L. Miller, and John H. Reif. "On Determining the Genus of a Graph in $O(V^{O(g)})$ Steps." In: *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*. Ed. by Michael J. Fischer, Richard A. DeMillo, Nancy A. Lynch, Walter A. Burkhard, and Alfred V. Aho. ACM, 1979, pp. 27–37. DOI: 10.1145/800135.804395.

[Fou74] Jean Claude Fournier. "Une Relation de Séparation entre Cocircuits d'un Matroide." In: *Journal of Combinatorial Theory. Series B* 16 (1974), pp. 181–190.

[Fox+16] Eli Fox-Epstein, Shay Mozes, Phitchaya Mangpo Phothilimthana, and Christian Sommer. "Short and simple cycle separators in planar graphs." In: *ACM Journal of Experimental Algorithmics* 21 (2016), Article 2.2, 24. ISSN: 1084-6654. DOI: 10.1145/2957318.

[FR82] H. de Fraysseix and P. Rosenstiehl. "A depth-first-search characterization of planarity." In: *Graph theory (Cambridge, 1981)*. Vol. 13. Ann. Discrete Math. North-Holland, Amsterdam-New York, 1982, pp. 75–80.

[FR85] H. de Fraysseix and P. Rosenstiehl. "A characterization of planar graphs by Trémaux orders." In: *Combinatorica. An International Journal of the János Bolyai Mathematical Society* 5.2 (1985), pp. 127–135. ISSN: 0209-9683. DOI: 10.1007/BF02579375.

[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 Commun.* 24.2 (2011), pp. 107–124. DOI: 10.3233/AIC-2011-0491.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness*. W. H. Freeman and Co., San Francisco, Calif., 1979, pp. x+338. ISBN: 0-7167-1045-5.

[GJ83] Michael R. Garey and David S. Johnson. "Crossing number is NP-complete." In: *SIAM Journal on Algebraic Discrete Methods* 4 (3 1983), pp. 312–316.

[GT87] Jonathan L. Gross and Thomas W. Tucker. *Topological graph theory*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., New York, 1987, pp. xvi+351. ISBN: 0-471-04926-3.

[Gur16] Inc. Gurobi Optimization. *Gurobi Optimizer Reference Manual*. 2016. URL: http://www.gurobi.com.

[Hef91] L. Heffter. "Ueber das Problem der Nachbargebiete." In: *Mathematische Annalen* 38.4 (1891), pp. 477–508. ISSN: 0025-5831. DOI: 10.1007/BF01203357.

[Jae79] François Jaeger. "Interval matroids and graphs." In: *Discrete Mathematics* 27.3 (1979), pp. 331–336. ISSN: 0012-365X. DOI: 10.1016/0012-365X(79)90167-5.

[JM96] Michael Jünger and Petra Mutzel. "Maximum Planar Subgraphs and Nice Embeddings: Practical Layout Tools." In: *Algorithmica. An International Journal in Computer Science* 16.1 (1996), pp. 33–59. ISSN: 0178-4617. DOI: 10.1007/s004539900036.

[JMM95] Martin Juvan, Joze Marincek, and Bojan Mohar. "Embedding Graphs in the Torus in Linear Time." In: *Integer Programming and Combinatorial Optimization, 4th International IPCO Conference, Copenhagen, Denmark, May 29-31, 1995, Proceedings*. Ed. by Egon Balas and Jens Clausen. Vol. 920. Lecture Notes in Computer Science. Springer, 1995, pp. 360–363. ISBN: 3-540-59408-6. DOI: 10.1007/3-540-59408-6_64.

[JS09] Michael Jünger and Michael Schulz. "Intersection Graphs in Simultaneous Embedding with Fixed Edges." In: *Journal of Graph Algorithms and Applications* 13.2 (2009), pp. 205–218. ISSN: 1526-1719. DOI: 10.7155/jgaa.00184.

[Jun13] Dieter Jungnickel. *Graphs, networks and algorithms*. Fourth. Vol. 5. Algorithms and Computation in Mathematics. Springer, Heidelberg, 2013, pp. xx+675. DOI: 10.1007/978-3-642-32278-5.

[Kin92] Nancy G. Kinnersley. "The vertex separation number of a graph equals its pathwidth." In: *Information Processing Letters* 42.6 (1992), pp. 345–350. ISSN: 0020-0190. DOI: 10.1016/0020-0190(92)90234-M.

[KKR12] Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce Reed. "The disjoint paths problem in quadratic time." In: *Journal of Combinatorial Theory. Series B* 102.2 (2012), pp. 424–435. ISSN: 0095-8956. DOI: 10.1016/j.jctb.2011.07.004.

[KMR08] Ken-ichi Kawarabayashi, Bojan Mohar, and Bruce A. Reed. "A Simpler Linear Time Algorithm for Embedding Graphs into an Arbitrary Surface and the Genus of Graphs of Bounded Tree-Width." In: *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*. IEEE Computer Society, 2008, pp. 771–780. ISBN: 978-0-7695-3436-7. DOI: 10.1109/FOCS.2008.53.

[KMV00] T. Koch, A. Martin, and S. Voß. *SteinLib: An Updated Library on Steiner Tree Problems in Graphs*. Tech. rep. ZIB-Report 00-37. Takustr. 7, Berlin: Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2000. URL: http://elib.zib.de/steinlib.

[Kow11] Emmanuel Kowalski. *Expander graphs*. Lecture notes, ETH Zürich. https://people.math.ethz.ch/~kowalski/lecture-notes.html. 2011.

[KP15] Michal Kotrbčík and Tomaž Pisanski. "Genus of the cartesian product of triangles." In: *Electronic Journal of Combinatorics* 22.4 (2015), Paper 4.2, 20. ISSN: 1077-8926.

[KR96] Jennifer Keir and Bruce Richter. "Walks through every edge exactly twice. II." In: *Journal of Graph Theory* 21.3 (1996), pp. 301–309. ISSN: 0364-9024. DOI: 10.1002/(SICI)1097-0118(199603)21:3<301::AID-JGT4>3.3.CO;2-S.

[KS15] Ken-ichi Kawarabayashi and Anastasios Sidiropoulos. "Beyond the Euler Characteristic: Approximating the Genus of General Graphs." In: *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*. Ed. by Rocco A. Servedio and Ronitt Rubinfeld. ACM, 2015, pp. 675–682. ISBN: 978-1-4503-3536-2. DOI: 10.1145/2746539.2746583.

[KS93] Ephraim Korach and Nir Solel. "Tree-width, path-width, and cutwidth." In: *Discrete Applied Mathematics. The Journal of Combinatorial Algorithms, Informatics and Computational Sciences* 43.1 (1993), pp. 97–101. ISSN: 0166-218X. DOI: 10.1016/0166-218X(93)90171-J.

[Kur30] Kazimierz Kuratowski. "Sur le problème des courbes gauches en topologie." In: *Fundamenta Mathematicae* 15 (1930), pp. 271–283.

[KV12] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms.* Fifth. Vol. 21. Algorithms and Combinatorics. Springer, Heidelberg, 2012, pp. xx+659. ISBN: 978-3-642-24487-2. DOI: 10.1007/978-3-642-24488-9.

[LG79] P. C. Liu and R. C. Geldmacher. "On the deletion of nonplanar edges of a graph." In: *Proceedings of the Tenth Southeastern Conference on Combinatorics, Graph Theory and Computing (Florida Atlantic Univ., Boca Raton, Fla., 1979).* Congress. Numer., XXIII–XXIV. Utilitas Math., Winnipeg, Man., 1979, pp. 727–738.

[LH82] C. H. C. Little and D. A. Holton. "Rings of bonds in graphs." In: *Journal of Combinatorial Theory. Series B* 33.1 (1982), pp. 1–6. ISSN: 0095-8956. DOI: 10.1016/0097-3165(82)90074-7.

[LH85] C. H. C. Little and D. A. Holton. "No graph has a maximal 3-ring of bonds." In: *Journal of Combinatorial Theory. Series B* 38.2 (1985), pp. 139–142. ISSN: 0095-8956. DOI: 10.1016/0095-8956(85)90079-6.

[Lie01] Annegret Liebers. "Planarizing Graphs - A Survey and Annotated Bibliography." In: *Journal of Graph Algorithms and Applications* 5.1 (2001), pp. 1–74. DOI: 10.7155/jgaa.00032.

[Lip+16] M. Lipton, E. Mackall, T. W. Mattman, M. Pierce, S. Robinson, J. Thomas, and I. Weinschelbaum. "Six variations on a theme: almost planar graphs." In: *ArXiv e-prints* (Aug. 2016). eprint: 1608.01973.

[Lit77] Charles H. C. Little. "On rings of circuits in planar graphs." In: (1977), 133–140. Lecture Notes in Math., Vol. 622.

[LS10] Charles H. C. Little and G. Sanjith. "Another characterisation of planar graphs." In: *Electronic Journal of Combinatorics* 17.1 (2010), Note 15, 7. ISSN: 1077-8926. URL: http://www.combinatorics.org/Volume_17/Abstracts/v17i1n15.html.

[Mah+17] Stephen J. Maher, Tobias Fischer, Tristan Gally, Gerald Gamrath, Ambros Gleixner, Robert Lion Gottwald, Gregor Hendel, Thorsten Koch, Marco E. Lübbecke, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Sebastian Schenker, Robert Schwarz, Felipe Serrano, Yuji Shinano, Dieter Weninger, Jonas T. Witt, and Jakob Witzig. *The SCIP Optimization Suite 4.0.* Tech. rep. 17-12. Takustr. 7, 14195 Berlin: ZIB, 2017.

[Man83] Anthony Mansfield. "Determining the thickness of graphs is NP-hard." In: *Mathematical Proceedings of the Cambridge Philosophical Society* 93 (1983), pp. 9–23.

[MBS12] Sarah McGinnis, Jeremy Berry, and E. J. Sanchez. *Subgroup Graphs of Non-Orientable and Oriented Genus One.* Presentation in the *Research Experiences for Undergraduates in Mathematics at Missouri State University* project supervised by Les Reid. 2012. URL: http://people.missouristate.edu/lesreid/reu/2012/PPT/jeremy_sarah_ej.pdf.

[MK11] Wendy J. Myrvold and William Kocay. "Errors in graph embedding algorithms." In: *Journal of Computer and System Sciences* 77.2 (2011), pp. 430–438. ISSN: 0022-0000. DOI: 10.1016/j.jcss.2010.06.002.

[Moh+85] Bojan Mohar, Tomaž Pisanski, Martin Škoviera, and Arthur T. White. "The cartesian product of three triangles can be embedded into a surface of genus 7." In: *Discrete Mathematics* 56.1 (1985), pp. 87–89. ISSN: 0012-365X. DOI: 10.1016/0012-365X(85)90197-9.

[Moh96] Bojan Mohar. "Embedding Graphs in an Arbitrary Surface in Linear Time." In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*. Ed. by Gary L. Miller. ACM, 1996, pp. 392–397. ISBN: 0-89791-785-5. DOI: 10.1145/237814.237986.

[MPW05] Dragan Marušič, Tomaž Pisanski, and Steve Wilson. "The genus of the GRAY graph is 7." In: *European Journal of Combinatorics* 26.3-4 (2005), pp. 377–385. ISSN: 0195-6698. DOI: 10.1016/j.ejc.2004.01.015.

[MT01] Bojan Mohar and Carsten Thomassen. *Graphs on Surfaces*. Johns Hopkins Studies in the Mathematical Sciences, 2001, pp. xii+291. ISBN: 0-8018-6689-8.

[Mut94] Petra Mutzel. "The maximum planar subgraph problem." PhD thesis. Köln University, 1994.

[Por08] Timo Poranen. "Two New Approximation Algorithms for the Maximum Planar Subgraph Problem." In: *Acta Cybernetica* 18.3 (2008), pp. 503–527. ISSN: 0324-721X.

[Rin55] Gerhard Ringel. "Über drei kombinatorische Probleme am n-dimensionalen Würfel und Würfelgitter." In: *Abh. Math. Sem. Univ. Hamburg* 20 (1955), pp. 10–19. ISSN: 0025-5858. DOI: 10.1007/BF02960735.

[Rin65a] Gerhard Ringel. "Das Geschlecht des vollständigen paaren Graphen." In: *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 28 (1965), pp. 139–150. ISSN: 0025-5858. DOI: 10.1007/BF02993245.

[Rin65b] Gerhard Ringel. "Der vollständige paare Graph auf nichtorientierbaren Flächen." In: *Journal für die Reine und Angewandte Mathematik* 220 (1965), pp. 88–93. ISSN: 0075-4102. DOI: 10.1515/crll.1965.220.88.

[Rin74] Gerhard Ringel. *Map Color Theorem*. Die Grundlehren der mathematischen Wissenschaften, Band 209. Springer-Verlag, New York-Heidelberg, 1974, pp. xii+191.

[RS83] Neil Robertson and P. D. Seymour. "Graph minors. I. Excluding a forest." In: *Journal of Combinatorial Theory. Series B* 35.1 (1983), pp. 39–61. ISSN: 0095-8956. DOI: 10.1016/0095-8956(83)90079-5.

[Sch12] Peter Schmidt. "Algoritmické vlastnosti vnorení grafov do plôch." in Slovak: *Algorithmic properties of embeddings of graphs into surfaces*. B.Sc. thesis. Comenius University, 2012.

[Sch14] Marcus Schaefer. "The Graph Crossing Number and its Variants: A Survey." In: *The Electronic Journal of Combinatorics* Dynamic Survey 21 (2014).

[Sch89] Walter Schnyder. "Planar Graphs and Poset Dimension." In: *Order. A Journal on the Theory of Ordered Sets and its Applications* 5.4 (1989), pp. 323–343. ISSN: 0167-8094. DOI: 10.1007/BF00353652.

[Sch90] Walter Schnyder. "Embedding Planar Graphs on the Grid." In: *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms, 22-24 January 1990, San Francisco, California*. Ed. by David S. Johnson. SIAM, 1990, pp. 138–148. ISBN: 0-89871-251-3. URL: http://dl.acm.org/citation.cfm?id=320176.320191.

[SW99] A. Steger and N. C. Wormald. "Generating random regular graphs quickly." In: *Combinatorics, Probability and Computing* 8.4 (1999). Random graphs and combinatorial structures (Oberwolfach, 1997), pp. 377–396. ISSN: 0963-5483. DOI: 10.1017/S0963548399003867.

[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.

[TDB88] Roberto Tamassia, Giuseppe Di Battista, and Carlo Batini. "Automatic graph drawing and readability of diagrams." In: *IEEE Trans. Systems, Man, and Cybernetics* 18.1 (1988), pp. 61–79. DOI: 10.1109/21.87055.

[Tho80] Carsten Thomassen. "Planarity and Duality of Finite and Infinite Graphs." In: *Journal of Combinatorial Theory. Series B* 29.2 (1980), pp. 244–271. ISSN: 0095-8956. DOI: 10.1016/0095-8956(80)90083-0.

[Tho89] Carsten Thomassen. "The Graph Genus Problem Is NP-Complete." In: *Journal of Algorithms. Cognition, Informatics and Logic* 10.4 (1989), pp. 568–576. ISSN: 0196-6774. DOI: 10.1016/0196-6774(89)90006-0.

[Tho97] Carsten Thomassen. "The Genus Problem for Cubic Graphs." In: *Journal of Combinatorial Theory. Series B* 69.1 (1997), pp. 52–58. ISSN: 0095-8956. DOI: 10.1006/jctb.1996.1721.

[Tut59] William Thomas Tutte. "Matroids and graphs." In: *Transactions of the American Mathematical Society* 90 (1959), pp. 527–552. ISSN: 0002-9947. DOI: 10.2307/1993185.

[Tut63] William Thomas Tutte. "How to draw a graph." In: *Proceedings of the London Mathematical Society. Third Series* 13 (1963), pp. 743–767. ISSN: 0024-6115. DOI: 10.1112/plms/s3-13.1.743.

[Wag37] Klaus Wagner. "Über eine Eigenschaft der ebenen Komplexe." In: *Mathematische Annalen* 114.1 (1937), pp. 570–590. ISSN: 0025-5831. DOI: 10.1007/BF01594196.

[Whi84] Arthur T. White. *Graphs, groups and surfaces*. Second. Vol. 8. North-Holland Mathematics Studies. North-Holland Publishing Co., Amsterdam, 1984, pp. xiii+314. ISBN: 0-444-87643-x.

[Wil93] S. Gill Williamson. "Canonical forms for cycles in bridge graphs." In: *Linear and Multilinear Algebra* 34.3-4 (1993), pp. 301–341. ISSN: 0308-1087. DOI: 10.1080/03081089308818229.

[You63] John William Theodore Youngs. "Minimal Imbeddings and the Genus of a Graph." In: *J. Math. Mech.* 12.2 (1963), pp. 303–315.