Cycles to the Rescue! Novel Constraints to Compute Maximum Planar Subgraphs Fast

Markus Chimani
Theoretical Computer Science, Osnabrück University, Germany
markus.chimani@uni-osnabrueck.de
https://orcid.org/0000-0002-4681-5550

Tilo Wiedera
Theoretical Computer Science, Osnabrück University, Germany
tilo.wiedera@uni-osnabrueck.de
https://orcid.org/0000-0002-5923-4114

Abstract

The NP-hard *Maximum Planar Subgraph* problem asks for a planar subgraph H of a given graph G such that H has maximum edge cardinality. For more than two decades, the only known non-trivial exact algorithm was based on integer linear programming and Kuratowski's famous planarity criterion. We build upon this approach and present new constraint classes – together with a lifting of the polyhedron – to obtain provably stronger LP-relaxations, and in turn faster algorithms in practice. The new constraints take Euler's polyhedron formula as a starting point and combine it with considering cycles in G. This paper discusses both the theoretical as well as the practical sides of this strengthening.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization, Mathematics of computing → Graph theory, Theory of computation → Linear programming

Keywords and phrases algorithm engineering, graph algorithms, integer linear programming, maximum planar subgraph

Digital Object Identifier 10.4230/LIPIcs.ESA.2018.19

Related Version `https://arxiv.org/abs/1806.08283`

Funding Supported by the German Research Foundation (DFG) project CH 897/2-1.

1 Introduction

The NP-hard *Maximum Planar Subgraph* (MPS) problem is an established question in graph theory, already discussed in the classical textbook by Garey and Johnson [14, 20]. Given a graph G, we ask for a largest subset $F \subseteq E(G)$ of edges such that F induces a planar graph. By contrast, the closely related *maximal* planar subgraph problem asks for a set of edges that we cannot extend without violating planarity and is trivially solvable in polynomial time. The inverse measure of MPS that counts the minimum number of edges that must be removed to obtain a planar subgraph, is called the *skewness* of G and denoted by $skew(G)$.

There are several reasons why this problem has received a good deal of attention: Graph theoretically, skewness is a very natural and common measure of non-planarity (like crossing number or genus). Algorithmically, finding a large planar subgraph is central to the planarization method [1, 5] that is heavily used in graph drawing: one starts with a large (favorably maximum) planar subgraph and re-inserts the deleted edges, typically to obtain a low number of overall crossings. In fact, this gives an approximation of the crossing number

with ratio roughly $\mathcal{O}\big(\Delta \cdot skew(G)\big)$ [9], where Δ is the maximum node degree. Furthermore, several graph problems become easier when the input's skewness is small or constant. E.g., we can compute a maximum flow in time $\mathcal{O}\big(skew(G)^3 \cdot |V(G)| \log |V(G)|\big)$ [16][1] – the same runtime complexity as on planar graphs if the skewness is constant.

There are several practical heuristic approaches to tackle the problem [10]. However, MPS is MaxSNP-hard, i.e., there is an upper bound < 1 on the obtainable approximation ratio unless $P = NP$ [2], and there are further limits known for specific algorithmic approaches [3,7]. Already a spanning tree gives an approximation ratio of 1/3, the best known ratio is 4/9 [2], and only recently a practical 13/33-approximation algorithm emerged [3].

Considering exact algorithms, options are scarce. Over two decades ago, an integer linear program based on Kuratowski's characterization of planarity was introduced in [23], which remained the only non-trivial exact algorithm. Only very recently, [8] showed the existence of potentially feasible alternatives to the Kuratowski-based approach, but the former still constitutes the practically by far most efficient (and theoretically most thoroughly explored) model. All known ILP models (including those discussed in this paper) can also directly solve the *weighted* MPS, i.e., identify the heaviest planar subgraph w.r.t. given edge weights.

Contribution. In this paper, we strengthen the Kuratowski model by introducing new constraints and supplementary variables, based on analyzing the cycles occurring in the solutions; see Section 3. In particular, we show in Section 3.2 that starting with the original Kuratowski model and considering cycles of growing lengths yields a natural hierarchy of ever stronger LP-relaxations. In Section 3.3, we establish additional constraint classes using our cycle variables to further strengthen the LP-relaxations, both theoretically and practically. We show the latter property in an experimental evaluation in Section 4. We skip the proofs of some lemmata, in which case we mark the lemma with '⋆'.

2 Preliminaries

Graph Notation. Our non-planar input graph is called G. Generally, we consider an undirected graph H, with nodes $V(H)$ and edges $E(H)$, which are cardinality-2 subsets of $V(H)$. We use $\delta_H(v)$ to denote all edges incident to node v in H and define the node degree $\deg_H(v) := |\delta_H(v)|$. If H is a subgraph of G, we write $H \subseteq G$. A (sub)graph is a *cycle* if it is connected and all its nodes have degree 2. The *girth* $\gamma(H)$ of H is the length of its smallest cycle. The union of two (non-disjoint) graphs H_1, H_2 is denoted by $H_1 \sqcup H_2 := (V(H_1) \cup V(H_2), E(H_1) \cup E(H_2))$. For $W \subseteq V(H)$ and $F \subseteq E(H)$ we define node- and edge-induced subgraphs $H[W] := (W, \{e \in E(H) \mid e \subseteq W\})$ and $H[F] := \big(\bigcup_{e \in F} e, F\big)$, respectively. We further use $H - e := H[E(H) \setminus \{e\}]$.

Given a planar drawing $\mathcal{D}$ of some planar graph H, the cyclic adjacency order around each node in $\mathcal{D}$ defines an *embedding* π of H. The disjoint regions bounded by edges in $\mathcal{D}$ correspond to the *faces* of π; the infinite region, bounded only on the inside, is called *outer face*. The *degree* $\deg(f)$ of any face f is the number of *half-edges* ("sides" of edges) that occur on the boundary of f; a bridge occurs twice on the same face.

Linear Programming. A *Linear Program* (LP) is a vector $c \in \mathbb{R}^d$ and a set of linear inequalities (*constraints*) that define a polyhedron P in $\mathbb{R}^d$; we ask for an element $x \in P$ that maximizes $c^\intercal x$. An *Integer Linear Program* (ILP) additionally requires the components

[1] [16] considers the crossing number; the algorithm trivially works also for the stronger parameter skewness.

of x to be integral. For a given problem, one can establish different ILPs, so-called *models*. To solve an ILP model, one uses branch-and-bound, where dual bounds are obtained from (fractional) solutions to the *LP-relaxation*, i.e., the ILP without the integrality requirements. Clearly, strong such LP-bounds are desired. We say a model N is *at least as strong* as a model M, if N's LP-relaxation gives no worse bounds than M's. We say N is *stronger* than M if, additionally, there is an instance where N gives a strictly better bound. If, in this case, N arises from M by adding some constraints C, we say C *strengthen* M.

It is often beneficial to consider only a relevant subset of constraints in the solving process, in particular when the class of constraints is (exponentially) large. The procedure is referred to as *separation*. We employ it on (fractional) LP-solutions for selected constraint classes.

Kuratowski Model (ε-Model). The following ILP is due to Mutzel [23]. Jünger and Mutzel showed that both constraint classes below form facets of the planar subgraph polytope [17]. We use solution variables $s_e \in \{0,1\}$ (for all $e \in E(G)$) that are 1 if and only if edge e is deleted, i.e., *not* in the planar subgraph. (In [23], equivalent variables $x_e := 1 - s_e$ are used.) The objective minimizes the skewness – thus maximizes the planar subgraph – and is given by

$$\min \sum\nolimits_{e \in E(G)} w(e)\, s_e.$$

Thereby, we may consider edge weights w; they are 1 in case of the traditional unweighted MPS problem. For a given subset $F \subseteq E(G)$ of edges, we define $s(F) := \sum_{e \in F} s_e$ as a shorthand. We can always use Euler's bound on the number of edges in planar graphs:

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

By Kuratowski's theorem [19], a graph is planar if and only if it neither contains a subdivision of a K_5 nor of a $K_{3,3}$. Hence, it suffices to ask for any member of the (exponentially large) set $\mathcal{K}(G)$ of all Kuratowski subdivisions that at least one of its edges is deleted:

$$s\big(E(K)\big) \geq 1 \qquad \forall K \in \mathcal{K}(G). \tag{2}$$

Clearly, (2) are too many constraints to use all explicitly. Instead, we identify a sufficient subset of constraints via a (heuristic) separation procedure: we round the fractional solution and obtain a graph that can be tested for planarity. If it is non-planar, we extract a Kuratowski subdivision. This method does neither guarantee to always find a violated constraint if there is any, nor that the identified subdivision in fact corresponds to a *violated* Kuratowski constraint. Still, since it has these guarantees on integral solutions, it suffices to obtain an exact algorithm. Over the years, the performance of this approach was improved by strong preprocessing [4], finding *multiple* Kuratowski subdivision in linear time [11], and strong primal heuristics [10]. We use all these identically in all considered algorithms.

The Kuratowski-model forms the basis of our extensions. As such, we denote it, without any of the below extensions, by 'ε-model'.

3 Stronger Constraints Based on Cycles

We now present new constraints for the planar subgraph polytope (or a lifted version thereof). All but the first class require the introduction of new variables based on cycles, leading to the *cycle model*. For each constraint class we first give some motivation and intuition for its feasibility, before discussing its technical details. We then describe – provided the class is large – separation routines that quickly identify violated constraints, and usually show that it strengthens our ILP model.

Generalized Euler Constraints. We know from [17] that inequality $|E(G)| \leq 2|V(G)| - 4$ is facet-defining for complete biconnected graphs. We are interested in a class of similar constraints for dense subgraphs with large girth. The following lemma is folklore:

▶ **Lemma 1.** A planar graph G has at most $\big(|V(G)| - 2\big)\gamma(G)/\big(\gamma(G) - 2\big)$ edges.

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

We can thus derive a feasible *generalized Euler constraint* for any subgraph $H \subseteq G$:

$$|E(H)| - s\big(E(H)\big) \leq \big(|V(H)| - 2\big)\gamma(H)/\big(\gamma(H) - 2\big) \qquad \forall H \subseteq G \tag{3}$$

We note that this bound can sometimes be improved: for constraints (3) to be satisfied with equality it is necessary that $V(H) \equiv 2 \pmod{\gamma(H) - 2}$ if $\gamma(H)$ is odd and $V(H) \equiv 2 \pmod{(\gamma(H) - 2)/2}$ otherwise [13]. However, we did not implement this in our algorithms.

▶ **Lemma 2.*** The generalized Euler constraints (3) strengthen the ε-model.

Proof sketch. $K_{3,3,1}$ contains a $K_{3,4}$ that prohibits the otherwise feasible solution 3/2. ◀

We separate constraints (3) heuristically by seeking dense, high-girth subgraphs using two different methods. First, using the current fractional solution, we assign weight $1 - s_e$ to each edge e and approximate a maximum cut [22, Section 6.3], obtaining a girth-4 subgraph. If (after postprocessing, see below) this does not yield a violated constraint, we try a second method: We set a target girth μ and iteratively add edges in ascending order of their LP-value to an initially empty graph, while updating the shortest paths between all node pairs. Upon adding an edge e, we check whether e would create a cycle of length $< \mu$, in which case we discard e instead. We may repeat this process for different values of μ. After each of the above attempts, we apply a postprocessing: Let H denote a girth-μ subgraph. The *contribution* of a node $v \in V(H)$ is defined by $|\delta_H(v)| - \sum_{e \in \delta_H(v)} s_e - \mu/(\mu - 2)$. We iteratively remove nodes with negative contribution from H. In particular, this will remove all degree-1 nodes.

3.1 Cycle Model

We now want to bound the number of edges in the planar subgraph by the number of its small faces. Even though compelling from a theoretical standpoint, it is infeasible to generate all potential faces of all planar subgraphs of a given graph (already for bounded length). However, we know that traversing the border of any face of a spanning subgraph H traverses at least one cycle if H is not a tree. We will relate the number of small faces in any planar subgraph of a graph G to the number of small cycles in G. One may also view this as a way to further generalize Euler constraints: many – in particular sparse – graphs have low girth only due to very few cycles of small length.

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

▶ **Lemma 3.** For every connected, planar but non-outerplanar subgraph H of G, there exists an embedding of H such that we can assign a unique cycle α to every face f where all edges of α occur on the boundary of f.

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

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

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

$$m = 3n - 6 - \sum_{d=3}^{2m} (d-3) f_{=d}. \tag{4}$$

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

Let $\mathcal{C}_d(G)$ denote all cycles of length d in G. We set $D \geq 3$ to the *maximum cycle length* that we want to investigate; this parameter will control the number of additionally generated variables. Let $\mathcal{C}_{\leq D}(G)$ denote the set of cycles with length at most D. For every cycle $\alpha \in \mathcal{C}_{\leq D}(G)$ we generate a variable $c_\alpha \in \{0, 1\}$.[2] We force such a variable to 0 if any edge of the respective cycle is removed and allow at most two cycles per edge in the MPS:

$$\sum\nolimits_{\alpha \in \mathcal{C}_{\leq D}(G)\colon e \in E(\alpha)} c_\alpha \leq 2\,(1 - s_e) \qquad \forall e \in E(G) \tag{5}$$

Note that constraints (5) resemble the requirement for each edge to appear in at most two faces (subject to Lemma 3). We discuss its correctness below. Let $c(d) := \sum_{\alpha \in \mathcal{C}_d(G)} c_\alpha$.

▶ **Lemma 5.** For every connected, planar but non-outerplanar subgraph H of G, there exists an embedding π of H and a feasible assignment w.r.t. (5) of cycle variables such that for each $d \leq D$ the number $f_{=d}$ of faces with degree d in π is bounded from above by

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

Proof. We assign cycle variables following the proof of Lemma 3. Hence, there is a unique cycle variable assigned to each face such that the length of the cycle is at most the degree of its face. The variable assignment is feasible since we pick only edges contained in H and pick at most two cycles incident with any such edge. ◀

▶ **Theorem 6.** *For any maximum planar subgraph of a graph G on n nodes and m edges there exists a feasible variable assignment that satisfies* (5) *and the* cycle constraint

$$(D-1)\big(m - s(E(G))\big) \leq (D+1)(n-2) + \sum_{d=3}^{D} (D+1-d)c(d). \tag{6}$$

[2] Intuitively, we want $c_\alpha = 1$ if and only if α is (part of) a face, see below for details. In terms of correctness, we need not but can actively force these variables to be binary, cf. Section 4.

Proof. Starting with (4) on any connected, planar subgraph of G that has $m - s(E(G))$ edges, we relax the equality by using the same coefficient for all faces of large degree as in

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

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

$$(D-1)\big(m - s(E(G))\big) \leq (D+1)(n-2) + \sum_{d=3}^{D}(D+1-d)f_{=d}. \tag{7}$$

The claimed cycle constraint is finally obtained by applying Lemma 5 to iteratively replace $f_{=D'}$ for $D' = D, D-1, \ldots, 4, 3$ by the upper bound (note that $f_{<3} = 0$), as sketched below for the (generalized, iteratively re-appearing) rightmost summand of (7):

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

3.2 Relaxations and D-Hierarchy

We now turn our attention to LP-relaxations of the cycle model. We show that there is a hierarchy of gradually stronger LPs induced by the maximum cycle length D. Let the *cycle model* CM_D consist of the ε-model, the cycle variables for cycle lengths up to D, and the corresponding constraints (5),(6). If $D = 2$ were allowed, CM_2 would be exactly the ε-model.

▶ **Lemma 7.** For any solution to the relaxation of CM_D, it holds that

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

Proof. Assume the contrary, $\sum_{d=3}^{D}(d-2)c(d) > 2n-4$. It follows that $\sum_{d=3}^{D} dc(d) > 2n-4+2\sum_{d=3}^{D}c(d)$ and hence $m - s(E(G)) > n-2+\sum_{d=3}^{D}c(d)$ by the sum of constraints (5). Plugging this bound on the number of edges into the cycle constraint (6), we obtain $\sum_{d=3}^{D}(d-2)c(d) < 2n-4$, a contradiction. ◀

It is not immediately clear, that decreasing the maximum cycle length maintains LP-feasibility, as some variables are removed and the cycle constraint is replaced. By employing Lemma 7, we can show the following fact.

▶ **Lemma 8.*** Model CM_{D+1} is at least as strong as CM_D.

▶ **Lemma 9.** Model CM_{D+1} is stronger than CM_D.

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

Since K_k^μ has girth μ there can be at most $(|V(K_k^\mu)| - 2)\mu/(\mu-2)$ edges in any planar subgraph. As there are no cycle variables, the cycle constraint (6) approaches this value from above for increasing D. Any feasible solution that tightly satisfies the cycle constraint is an optimal one. The Kuratowski constraints (2) on the other hand are already satisfied by deleting each edge partially with $s_e = 1/(9\xi)$ $\forall e \in E(K_n^\mu)$, since each subdivision requires at least 9ξ edges, still allowing LP-solutions with value $k(k-1)/18$. ◀

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

3.3 Strengthening the Cycle Model

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

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

$$t_{vw} + t_{wv} \leq 1 - s_{\{v,w\}} \qquad \forall \{v,w\} \in E \tag{8}$$

$$\sum\nolimits_{w \in N(v)} t_{vw} \geq t_v \qquad \forall v \in V(G) \tag{9}$$

$$t_v + \deg_G(v) - \sum\nolimits_{w \in N(v)} t_{wv} - \sum\nolimits_{w \in N(v)} s_{vw} \geq 2 \qquad \forall v \in V(G) \tag{10}$$

Constraints (8) allow at most one tree-arc for any edge and none for deleted edges. We force tree nodes to propagate along one outgoing arc by constraints (9). Finally, constraints (10) label degree-1-nodes and nodes where all (but one) neighbor is labeled. Now, we may subtract $\sum_{v \in V(G)} t_v$ nodes (and the same number of edges) from (6) to obtain a stronger bound:

▶ **Corollary 10.** The *extended cycle constraint*, given below, is feasible.

$$(D-1)\big(m - s(E(G))\big) \leq (D+1)(n-2) + \sum\nolimits_{d=3}^{D} (D+1-d)c(d) - 2\sum\nolimits_{v \in V(G)} t_v \tag{11}$$

Alternatively, we may use a less sophisticated approach that does not model propagation but labels only degree-1-nodes. In this case, it suffices to add variables $t_v \in \{0,1\}$, $\forall v \in V(G)$, and constraints (10), assuming $\sum_{w \in N(v)} t_{wv} = 0$.

▶ **Lemma 11.** ⋆ The pseudo-tree extension, i.e., constraints (8)–(11) together with the t-variables, strengthens CM_3. This already holds for the approach without propagation.

Proof sketch. We use the graph given in Fig. 1a: any MPS of it has a degree-1 node. ◀

All following constraint classes deal with excluding combinations of cycles and paths that either induce non-planarity, or result in cycle-variables not assignable to any face in the planar subgraph (Lemma 5). They are independent of but compatible with the pseudo-tree extension.

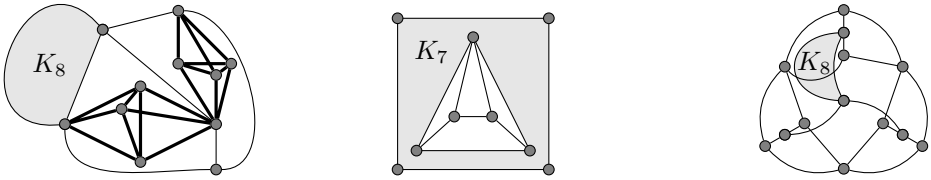
Cycle-Edge Constraints. Considering integral solutions and constraints (5), a cycle cannot be picked if any of its edges is deleted. W.r.t. fractional solutions we can additionally require

$$s_e + c_\alpha \leq 1 \qquad \forall \alpha \in \mathcal{C}_{\leq D}, e \in E(\alpha). \tag{12}$$

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

▶ **Lemma 12.** ⋆ The cycle-edge constraints (12) strengthen CM_3.

Proof sketch. Use a graph (Fig. 1b) that has 3 edges each incident to only 1 triangle. ◀



(a) Pseudo-Tree, Lemma 11 **(b)** Cycle-Edge, Lemma 12 **(c)** Two-Cycles-Path, Lemma 14

Figure 1 Graphs in strength proofs. Bold edges in **(a)** have large weight (or are edge bundles).

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

▶ **Lemma 13.** The two-cycles-path constraints, given below, are feasible.

$$s\big(E(p)\big) \geq c_\alpha + c_\beta - 1 \qquad \forall \alpha, \beta \in \mathcal{C}_{\leq D}; p \in \psi(\alpha, \beta) \tag{13}$$

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

▶ **Lemma 14.** ⋆ The two-cycles-path constraints (13) strengthen CM_4.

Proof sketch. We use the graph of Fig. 1c as input. ◀

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

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

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

▶ **Lemma 15.** The cycle-two-paths constraints, given below, are feasible.

$$s\big(E(p_1 \sqcup p_2)\big) \geq c_\alpha \qquad \forall \alpha \in \mathcal{C}_{\leq D}, \forall \text{ conflicting paths } p_1, p_2 \text{ w.r.t. } \alpha \tag{14}$$

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

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

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

▶ **Lemma 16.** The Kuratowski-cycle constraints, given below, are feasible.

$$s(\{e \in E(K) \mid \forall \alpha \in C : e \notin E(\alpha)\}) \geq \sum\nolimits_{\alpha \in \mathcal{C}} c_\alpha + 1 - |C| \qquad \forall K \in \mathcal{K}, C \subseteq \mathcal{C}_{\leq D} \tag{15}$$

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

▶ **Lemma 17.*** The Kuratowski-cycle constraints (15) strengthen CM_4.

Proof sketch. We use the circulant on 16 nodes with jumps 1, 2, and 8 as input. ◀

For separation, we identify a Kuratowski subdivision K as for (2). We collect the set S of cycles with LP-value > 0 incident with K. For each cycle in S, we compute its *gain*, i.e., the increase in violation (or decrease in slack) when adding that cycle to C. While there are cycles with positive gain, we continue adding a cycle of S with maximal gain to C.

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

▶ **Lemma 18.** The cycle-clique constraints, given below, are feasible.

$$\sum\nolimits_{\alpha \in C} c_\alpha \leq 1 \qquad \forall C \subseteq \mathcal{C}_{\leq D} \text{ s.t. all cycles in } C \text{ are pairwise conflicting} \tag{16}$$

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

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

4 Experiments

All algorithms are implemented in `C++`, compiled with GCC 6.3.0, and use the OGDF (snapshot 2017-07-23) [6]. We use SCIP 4.0.1 for solving ILPs with CPLEX 12.7.1 as the underlying LP solver [21]. Each MPS-computation uses a single physical core of a Xeon Gold 6134 CPU (3.2 GHz) with a memory speed of 2666 MHz. We employ a time limit of 20 minutes and a memory limit of 8 GB per computation. Our instances and results, giving runtime and skewness (if solved), are available for download at `http://tcs.uos.de/research/mps`.

Instances and Algorithms. Analogously to the study [8], we consider three established real-world benchmark sets: *Rome* [12], *North* [24], and (a subset of) *SteinLib* [18]. We know from [7, 8] that random regular graphs (which are *expander graphs* with high probability) are especially hard to solve exactly. We use the same such instances as [8], but only consider graphs with ≤ 100 nodes, as no known exact algorithm solves larger instances. There are 20 graphs for each parameterization $(|V(G)|, \Delta) \in \{10, 20, 30, 50, 100\} \times \{4, 6, 10, 20, 40\}$, where $\Delta < |V(G)|$ is the node-degree. For tuning of ε (e.g., heap size in separation) we rely on the values identified in [15]. We use the notation below to encode algorithmic choices.

ε	Do not use any extensions but the basic Kuratowski algorithm [23].
e	Separate generalized Euler constraints (3).
c{r}	Add cycle constraints (5), (6), and variables with the minimal value for D such that there are variables for at least $100r$ cycles.
t{0|1}	Use the pseudo-tree extension (8)–(11) with (=t1) or without (=t0) propagation.
i	Enforce integrality of variables for cycles and pseudo-trees.
s	Separate cycle-edge constraints (12).
w{0|1}	Separate two-cycles-path constraints (13) using *combined* (=w0) or *separate* (=w1) approach. Also enables separation on cycle-clique constraints (16) for 2-cliques.
k	Separate Kuratowski-cycle constraints (15).
q	Separate cycle-clique constraints (16).

Note that instead of providing D explicitly, we specify a minimum number r' of cycle variables to be generated. We increment D while there are less than r' cycle variables.

Results. Table 1 shows the success rates (percentage of instances solved to proven optimality) and average runtime per instance of our algorithmic variants. For non-solved instances we assume the maximum runtime of 20 minutes – average runtimes are thus comparable only for algorithms that achieve roughly equal success rates. We group the variants by the number of used extensions and highlight variants that dominate their group in bold. The latter informs our choice of which variants to consider in the next group.

The separation of generalized Euler constraints is clearly beneficial only on the North graphs, but even there its improvements are marginal when compared to the cycle-based approach. The latter works very well in practice, for all instance sets. In particular (cf. Fig. 2), on *Rome* it allows us for the first time to compute the skewness of *all* instances. Using variant *c10 t0 i w0*, we are able to solve all but `grafo10958.98.lgr` within the 20 minute time frame; this last instance required 103 minutes. *North* still contains instances too hard to solve exactly (even when increasing runtime to a few days and memory to 32GB). Nonetheless, we now solve 3/4 of the previously unsolved *North* graphs within our strict limits. The second group of variants in Table 1 demonstrates that all of our extensions of the cycle model, in particular the pseudo-tree approach, improve upon success rate and runtime on all instance sets when applied to *c10*. As shown in the lower sections of the table, this does not always apply when comparing models that simultaneously use multiple extensions.

Table 2 details the relative improvement for each of the three most promising algorithm configurations over the state-of-the-art ε-model. We provide the success rate for the instances not solved by ε and give the average relative speed-up (i.e., the runtime of ε divided by that of variant X) over the instances solved by both ε and X. This common set is exactly those solved by ε, except for a single ε-solved *North*-instance not solved by *c10*. On *Rome*, the pure cycle model *c10* without any further extensions achieves the best speed-up; for the seemingly harder other instance sets, more sophisticated variants are worthwhile. Fig. 2 underlines that the success rate of the algorithms is strongly correlated to the instance's skewness.

Table 1 Overview of performance for algorithmic variants: success rate and avg. runtime.

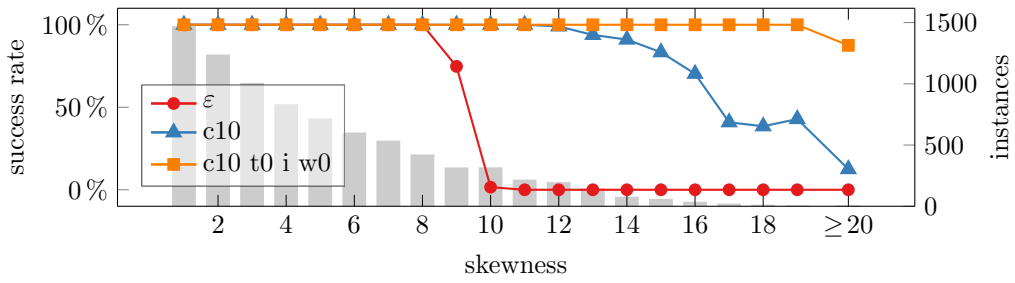
variant	Rome		North		Expanders		SteinLib	
	succ. [%]	time [s]	succ. [%]	time [s]	succ. [%]	time [s]	succ. [%]	time [s]
ε	85.71	198.42	73.76	325.31	34.74	800.38	9.52	1 085.94
e	85.56	199.41	77.78	273.29	35.00	803.91	9.52	1 085.93
c5	98.91	21.60	84.40	201.42	53.95	567.52	31.43	859.40
c10	**99.14**	**18.10**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

Table 2 Relative improvement over ε for selected algorithmic variants. We give the the success rate over the instances unsolved by ε, and the avg. runtime ratio over the commonly solved instances.

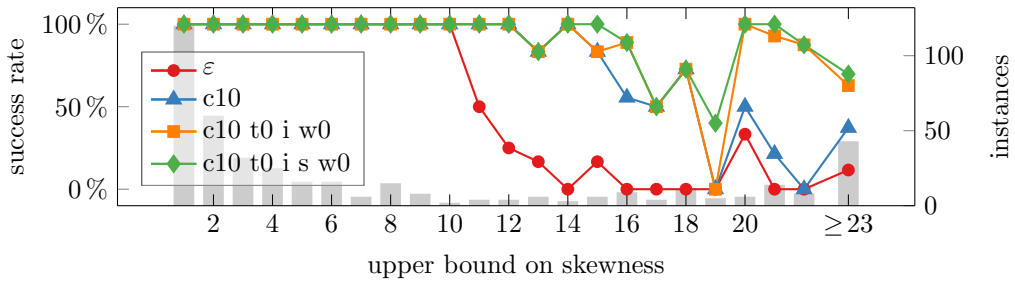
variant	Rome		North		Expanders		SteinLib	
	new [%]	speed-up	new [%]	speed-up	new [%]	speed-up	new [%]	speed-up
c10	93.98	**66.80**	42.34	21.45	30.24	13.96	25.26	**11.79**
c10 t0 i w0	**99.92**	60.85	72.07	28.59	34.27	12.68	31.58	6.79
c10 t0 i s w0	99.75	59.58	**78.38**	**34.03**	**37.90**	**23.21**	**32.63**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

Table 3 Average number of cycle variables and average values for maximum cycle length D.

variant		Rome		North		Expanders		SteinLib	
	min # var	D	# var	D	# var	D	# var	D	# var
c5	500	9.51	627	7.34	689	5.43	2 075	5.80	881
c10	1 000	10.51	1 168	8.01	1 213	5.73	2 816	6.64	3 658
c20	2 000	11.51	2 175	8.55	2 048	6.47	7 774	7.09	4 785



(a) Solved Rome graphs by skewness. *c10 t0 i w0* solves all but 1 skew-22-graph within 20min.



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

Figure 2 Detailed success rates for selected algorithmic variants.

Table 3 lists the average number of generated cycle variables and the respective average values for D. We mention that instances with high D values typically generate *few* cycle variables, close to the lower bound. However, there is a large deviation in the number of generated cycle variables in any fixed instance set: some graphs contain less than the requested number of cycles whereas others already contain roughly 10 000 triangles.

5 Conclusion and Open Questions

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

Several of our proofs show the new constraint class's strength w.r.t. a low-D cycle model. We conjecture that most classes remain strengthening for high D, but to prove this, one has to find and argue infinite families of graphs with the LP-properties of our currently hand-crafted proof graphs. Furthermore, it is natural to ask if and which of the new constraint classes form facets in the (lifted) planar subgraph polytope.

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

References

1 Carlo Batini, Maurizio Talamo, and Roberto Tamassia. Computer Aided Layout of Entity Relationship Diagrams. *J. Syst. Soft.*, 4(2-3):163–173, 1984. doi:10.1016/0164-1212(84)90006-2.

2 Gruia Călinescu, Cristina Gomes Fernandes, Ulrich Finkler, and Howard Karloff. A Better Approximation Algorithm for Finding Planar Subgraphs. *J. Alg. in Cognition, Informatics and Logic*, 27(2):269–302, 1998. doi:10.1006/jagm.1997.0920.

3 Parinya Chalermsook and Andreas Schmid. Finding Triangles for Maximum Planar Subgraphs. In Sheung-Hung Poon, Md. Saidur Rahman, and Hsu-Chun Yen, editors, *WALCOM: Algorithms and Computation, 11th International Conference and Workshops, WALCOM 2017, Hsinchu, Taiwan, March 29-31, 2017, Proceedings.*, volume 10167 of *Lecture Notes in Computer Science*, pages 373–384. Springer, 2017. doi:10.1007/978-3-319-53925-6_29.

4 Markus Chimani and Carsten Gutwenger. Non-planar core reduction of graphs. *Discrete Mathematics*, 309(7):1838–1855, 2009. doi:10.1016/j.disc.2007.12.078.

5 Markus Chimani and Carsten Gutwenger. Advances in the Planarization Method: Effective Multiple Edge Insertions. *J. Graph Algorithms Appl.*, 13(3):729–757, 2012. doi:10.7155/jgaa.00264.

6 Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W. Klau, Karsten Klein, and Petra Mutzel. The Open Graph Drawing Framework (OGDF). In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization*, pages 543–569. Chapman and Hall/CRC, 2013. URL: https://crcpress.com/Handbook-of-Graph-Drawing-and-Visualization/Tamassia/9781584884125.

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

8 Markus Chimani, Ivo Hedtke, and Tilo Wiedera. Exact Algorithms for the Maximum Planar Subgraph Problem: New Models and Experiments. In *17th International Symposium on Experimental Algorithms, SEA 2018, June 27-29, 2018, L'Aquila, Italy*, volume 103. LIPIcs, 2018. doi:10.4230/LIPIcs.SEA.2018.22.

9 Markus Chimani and Petr Hlinený. A tighter insertion-based approximation of the crossing number. *J. Comb. Optim.*, 33(4):1183–1225, 2017. doi:10.1007/s10878-016-0030-z.

10 Markus Chimani, Karsten Klein, and Tilo Wiedera. A Note on the Practicality of Maximal Planar Subgraph Algorithms. In Yifan Hu and Martin Nöllenburg, editors, *Proceedings of the 24th International Symposium on Graph Drawing and Network Visualization (GD 2016)*, volume abs/1609.02443. CoRR, 2016. doi:10.1007/978-3-319-50106-2_28.

11 Markus Chimani, Petra Mutzel, and Jens M. Schmidt. Efficient Extraction of Multiple Kuratowski Subdivisions. In Seok-Hee Hong, Takao Nishizeki, and Wu Quan, editors, *Graph Drawing, 15th International Symposium, GD 2007, Sydney, Australia, September 24-26, 2007. Revised Papers*, volume 4875 of *Lecture Notes in Computer Science*, pages 159–170. Springer, 2007. doi:10.1007/978-3-540-77537-9_17.

12 Giuseppe Di Battista, Ashim Garg, Giuseppe Liotta, Roberto Tamassia, Emanuele Tassinari, and Francesco Vargiu. An experimental comparison of four graph drawing algorithms. *Computational Geometry. Theory and Applications*, 7(5-6):303–325, 1997. 11th ACM Symposium on Computational Geometry (Vancouver, BC, 1995). doi:10.1016/S0925-7721(96)00005-3.

13 Manuel Fernández, Nicholas Sieger, and Michael Tait. Maximal Planar Subgraphs of Fixed Girth in Ramdom Graphs. *CoRR*, 2018. arXiv:1706.06202.

14 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.

15 Ivo Hedtke. *Minimum Genus and Maximum Planar Subgraph: Exact Algorithms and General Limits of Approximation Algorithms.* PhD thesis, Osnabrück University, 2017. URL: `https://repositorium.ub.uos.de/handle/urn:nbn:de:gbv:700-2017082416212`.

16 Jan M. Hochstein and Karsten Weihe. Maximum s-t-flow with k crossings in $O(k^3 n \log n)$ time. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, SODA '07, pages 843–847, 2007.

17 Michael Jünger and Petra Mutzel. Maximum Planar Subgraphs and Nice Embeddings: Practical Layout Tools. *Algorithmica*, 16(1):33–59, 1996. `doi:10.1007/s004539900036`.

18 Thorsten Koch, Alexander Martin, and Stefan Voß. SteinLib: An updated library on steiner tree problems in graphs. Technical Report ZIB-Report 00-37, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustr. 7, Berlin, 2000. URL: `http://elib.zib.de/steinlib`.

19 Kazimierz Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamenta Mathematicae*, 15:271–283, 1930.

20 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, pages 727–738. Utilitas Math., Winnipeg, Man., 1979.

21 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. Technical Report 17-12, ZIB, Takustr. 7, 14195 Berlin, 2017.

22 Michael Mitzenmacher and Eli Upfal. *Probability and computing - randomized algorithms and probabilistic analysis.* Cambridge University Press, 2005.

23 Petra Mutzel. *The maximum planar subgraph problem.* PhD thesis, Köln University, 1994.

24 Stephen C. North. 5114 directed graphs, 1995. Manuscript.