A New Approach to Exact Crossing Minimization

Markus Chimani[1], Petra Mutzel[1], and Immanuel Bomze[2]

[1] Faculty of Computer Science, Dortmund University of Technology, Germany
{markus.chimani,petra.mutzel}@tu-dortmund.de
[2] Dep. of Statistics and Decision Support Systems, University of Vienna, Austria
immanuel.bomze@univie.ac.at

Abstract. The *crossing number* problem is to find the smallest number of edge crossings necessary when drawing a graph into the plane. Even-though the problem is NP-hard, we are interested in practically efficient algorithms to solve the problem to provable optimality. In this paper, we present a novel integer linear programming (ILP) formulation for the crossing number problem. The former formulation [4] had to transform the crossing number polytope into a higher-dimensional polytope. The key idea of our approach is to directly consider the natural crossing number polytope and cut it with multiple linear-ordering polytopes. This leads to a more compact formulation, both in terms of variables and constraints.
We describe a Branch-and-Cut algorithm, together with a combinatorial column generation scheme, in order to solve the crossing number problem to provable optimality. Our experiments show that the new approach is more effective than the old one, even when considering a heavily improved version of the former formulation (also presented in this paper). For the first time, we are able to solve graphs with a crossing number of up to 37.

1 Introduction

A *drawing* of a graph $G = (V, E)$ in the plane is a one-to-one mapping of each vertex to a point in $\mathbb{R}^2$, and each edge to a curve between its two endpoints. The curve is not allowed to contain other vertices than its two endpoints. A *crossing* is a common point of two curves, other than their endpoints. We forbid common points of more than two curves, other than their endpoints. The *crossing number* $\mathrm{cr}(G)$ is the smallest number of crossings in any drawing of G. The NP-hard problem of finding $\mathrm{cr}(G)$ has been widely studied in the literature – see [20] for an extensive bibliography – both from the graph theoretic, as well as the algorithmic point of view.

Recently, Buchheim et al. [4] presented the first exact algorithm to solve this problem to provable optimality, based on an integer linear programming (ILP) formulation: The central idea in all these formulations is to have a variable $x_{\{e,f\}}$ for each pair of edges $e, f \in E$, which is 1 if these edges cross, and zero otherwise.

The convex hull of the feasible points of x form the *crossing number polytope* $\mathcal{P}_{\text{cr}}$. Unfortunately, there is no known way to describe $\mathcal{P}_{\text{cr}}$ directly, as already checking if a given solution $\bar{x}$ is feasible – known as the *Realizability* problem – is NP-complete [15, 19]. If each edge is involved in only a single crossing, checking feasibility becomes simple: we can substitute each crossing by a *dummy vertex* of degree 4 and perform any planarity testing algorithm on the transformed graph. Hence the problem lies in edges e which are involved in multiple crossings, if we do not know the order of these crossings on e.

The formulation of [4] circumvents this problem by subdividing the graph such that each edge is replaced by a path of ℓ segments. Then, the formulation considers the *simple crossing number* instead, i.e., the smallest number of crossings in any drawing of G under the restriction that each edge-segment is involved in at most one crossing. Clearly, this solves the traditional crossing number problem on G if ℓ is large enough: since the optimal drawing of G might require all crossings to be on a single edge, we can select $\ell := \overline{\text{cr}}(G)$, some upper bound on the crossing number which may be obtained by a heuristic. Since $\text{cr}(G) = \mathcal{O}(|E|^2)$ and there are graphs with $\text{cr}(G) = \Omega(|E|^2)$, we obtain $\mathcal{O}(|E|^4)$ variables. We denote this formulation by SOCM, for *subdivision-based optimal crossing minimization*.

The enlarging of the input graph results in far too many variables to handle the problem efficiently, hence column generation schemes are proposed and compared in [8]: the therein presented *combinatorial column generation* – a scheme based on combinatorial and graph-theoretical arguments, rather than on algebraic concepts – offers a large improvement compared to traditional approaches based on reduced costs. Nonetheless, the approach, as presented in [8], was only suitable for relatively sparse graphs with roughly 70 nodes.

In this paper we present a competing ILP formulation based on *linear ordering* of crossings on any edge: we avoid the aforementioned graph expansion and require only $\mathcal{O}(|E|^3)$ instead of $\mathcal{O}(|E|^4)$ variables. We call this formulation OOCM, for *ordering-based optimal crossing minimization*. As the number of variables is still quite large, we furthermore present an efficient corresponding combinatorial column generation scheme.

From the polyhedral point of view, we can describe the situation as follows: checking the feasibility of a solution $\bar{x}$ is NP-complete and there is no known way to directly describe the feasible integer points of the polytope $\mathcal{P}_{\text{cr}}$. Hence, the SOCM formulation expands the input and considers the simpler polytope $\mathcal{P}_{\text{scr}}$ of the simple crossing number problem. In OOCM, we instead solve the problem directly in $\mathcal{P}_{\text{cr}}$, by cutting it with $O(|E|)$ many linear-ordering polytopes.

In the next section, we present the ILP formulation, while Section 3 describes the resulting Branch-and-Cut-and-Price algorithm and its sub-steps. In Section 4 we discuss extensions of OOCM for other types of crossing numbers and present recent improvements of SOCM which lead to improved performance compared to the results published in [4, 8]. Finally, in Section 5 we compare the improved SOCM implementation to the novel OOCM implementation by way of experiment.

2 The OOCM ILP formulation

It is a well-known fact that the crossing number of any graph is the sum of the crossing numbers of its biconnected components. Hence we can assume that the given graph G is at least 2-connected. Furthermore, we can confine ourselves to *simple* graphs, i.e., graphs without multi-edges or self-loops. While loops are irrelevant for the crossing number, we can get rid of multi-edges by introducing integer edge weights c. The crossing number can be obtained by counting $c_e \cdot c_f$ crossings for a crossing between the edges e and f. The need for these weights is further strengthened by the *non-planar core reduction* [7]: this preprocessing scheme shrinks a given 2-connected graph further without changing its crossing number, but introduces integer edge weights. Hence we will consider (G, c) as our input.

2.1 Variables and Linear Ordering

First, we orient all edges of G arbitrarily. For notational simplicity we continue to refer to the resulting graph as $G = (V, E)$. Let $\mathrm{E}^{\langle k \rangle} := \{(e_1, \ldots, e_k) \mid \forall\, 1 \leq i < j \leq k : e_i, e_j \in E \wedge e_i \neq e_j\}$ be the set of all ordered k-tuples of pairwise distinct edges. We model the order of the crossings directly via variables:

$$x_{\{e,f\}} \in \{0,1\} \quad \forall \{e,f\} \in \binom{E}{2}, \quad y_{e,f,g} \in \{0,1\} \quad \forall (e,f,g) \in \mathrm{E}^{\langle 3 \rangle} \tag{1}$$

A variable $x_{\{e,f\}}$ specifies whether or not the edges e and f cross. A variable $y_{e,f,g}$ is 1 if and only if both edges f and g cross e, and the crossing (e, f) is nearer to e's source node than the crossing (e, g). We say e is the *base* of the variable. The objective function of our ILP is then:

$$\min \sum_{\{e,f\} \in \binom{E}{2}} c_e \cdot c_f \cdot x_{\{e,f\}}$$

It is known that certain crossing-variables can be fixed to 0 as, e.g., there will never be crossings between adjacent edges. Any sensible implementation will ignore such variables.

Linear-Ordering Constraints. We define the set of *linear-order* (LO) constraints which ensure a consistent linear ordering over all edges:

$$x_{\{e,f\}} \geq y_{e,f,g}, \quad x_{\{e,g\}} \geq y_{e,f,g} \qquad \forall (e,f,g) \in \mathrm{E}^{\langle 3 \rangle} \tag{2}$$

$$1 + y_{e,f,g} + y_{e,g,f} \geq x_{\{e,f\}} + x_{\{e,g\}} \qquad \forall (e,f,g) \in \mathrm{E}^{\langle 3 \rangle} \tag{3}$$

$$y_{e,f,g} + y_{e,g,f} \leq 1 \qquad \forall (e,f,g) \in \mathrm{E}^{\langle 3 \rangle} \tag{4}$$

$$y_{e,f,g} + y_{e,g,h} + y_{e,h,f} \leq 2 \qquad \forall (e,f,g,h) \in \mathrm{E}^{\langle 4 \rangle} \tag{5}$$

We introduce *crossing-existence* constraints (2) which connect the x and y variables by ensuring that the x-vector specifies a crossing if the y-variables do. Vice

versa, the *order-existence* constraints (3) ensure that if x specifies two crossings on the same edge, the y-vector has to specify their order. The *mirror-order* constraints (4) guarantee that two crossings are uniquely ordered if they exist. Analogously, the *cyclic-order* constraints (5) ensure that the ordering is acyclic. A solution $(\bar{x}, \bar{y})$ which satisfies the LO-constraints is called *LO-feasible*. Since no two edges will ever cross more than once in any optimal solution, we have:

Proposition 1. *Let $\bar{x}$ be any optimal solution to the crossing number problem of any graph G. There exists an assignment $\bar{y}$ for the vector y such that $(\bar{x}, \bar{y})$ is LO-feasible.*

Checking feasibility. Let $(\bar{x}, \bar{y})$ be any integer LO-feasible solution. We replace each crossing in G by a dummy vertex. Since we know the intended order of these dummy vertices on each edge from the information in $(\bar{x}, \bar{y})$, the resulting graph is the *(partial) planarization* of G, which we denote by $G[\bar{x}, \bar{y}]$. We can check feasibility of $(\bar{x}, \bar{y})$ by testing $G[\bar{x}, \bar{y}]$ for planarity.

2.2 Kuratowski Constraints and Correctness of OCM

The final class of constraints required to fully describe the feasible points of our ILP are the *Kuratowski-constraints*. They guarantee that a computed integer LO-feasible solution $(\bar{x}, \bar{y})$ corresponds to a feasible planarization, i.e., $G[\bar{x}, \bar{y}]$ is planar: the well-known theorem by Kuratowski [16] states that a graph is planar if and only if it contains no *Kuratowski-subdivision* as a subgraph. A Kuratowski-subdivision results from subdividing the edges of a K_5 (complete graph on 5 nodes) or $K_{3,3}$ (complete bipartite graph with 3 nodes per partition) into paths of length at least 1, called *Kuratowski-paths*. The original nodes not obtained by the subdivision of the edges are called *Kuratowski-nodes*.

For any Kuratowski-subdivision K, we require at least one crossing between the edges of K. Such a subdivision might not be a subgraph of the original graph G, but might occur only in a partial planarization $G[\bar{x}, \bar{y}]$ for some integer LO-feasible solution $(\bar{x}, \bar{y})$.

For SOCM we simply use the crossings in such a planarization to "turn off" Kuratowski-constraints that are only valid if these crossings are selected [4]. The drawback is that these constraints are specifically tied to certain crossings, say between the edges e and f_1. This unavoidably leads to a multitude of very similar constraints, where, e.g., f_1 is replaced by another edge f_2, but f_1 and f_2 were created by the graph enlargement and correspond to the same original edge.

We cannot reuse such a simple approach straight-forwardly for OOCM. But now the additional effort is compensated for by constraints which correspond to a whole class of similar Kuratowski-constraints in SOCM. Let $(\bar{x}, \bar{y})$ be an integer LO-feasible solution, and let K be a Kuratowski-subdivision in $G[\bar{x}, \bar{y}]$. We define $\mathcal{Z}_K[\bar{x}, \bar{y}]$ as the set of crossings induced by $(\bar{x}, \bar{y})$ whose dummy nodes form integral parts of K: any $\{e, f\} \in \mathcal{Z}_K[\bar{x}, \bar{y}]$ either induces a Kuratowski-node or there exist a segment e' of e, a segment f' of f, and a Kuratowski-path which contains $\langle e', f' \rangle$ as a subpath. We can then define the *crossing shadow* $(\mathcal{X}_K[\bar{x}, \bar{y}], \mathcal{Y}_K[\bar{x}, \bar{y}])$ as a pair of sets as follows:

$\mathcal{Y}_K[\bar{x},\bar{y}] := \{(e,f,g) \in \mathrm{E}^{\langle 3\rangle} \mid \{e,f\},\{e,g\} \in \mathcal{Z}_K[\bar{x},\bar{y}] \wedge \bar{y}_{e,f,g} = 1 \wedge \nexists\{e,h\} \in \mathcal{Z}_K : \bar{y}_{e,f,h} = \bar{y}_{e,h,g} = 1\}$, i.e., a triple (e,f,g) is in $\mathcal{Y}_K[\bar{x},\bar{y}]$, if no other edge crosses e between f and g. Thus $\mathcal{Y}_K[\bar{x},\bar{y}]$ contains a minimal description of all crossings and their orderings in K, except for crossings of two edges, both not involved in multiple crossings; these are collected in the following set:
$\mathcal{X}_K[\bar{x},\bar{y}] := \{\{e,f\} \in \mathcal{Z}_K[\bar{x},\bar{y}] \mid \forall g \in E : \{(e,f,g),(e,g,f),(f,e,g),(f,g,e)\} \cap \mathcal{Y}_K[\bar{x},\bar{y}] = \emptyset\}$, i.e., all *singular crossings* in K not contained in $\mathcal{Y}_K[\bar{x},\bar{y}]$.

Proposition 2. *For each integer LO-feasible solution $(\bar{x},\bar{y})$ and each Kuratowski-subdivision K in $G[\bar{x},\bar{y}]$ we have: the partial planarization of G only realizing the crossings (and their order) as defined by the crossing shadow, contains K as a Kuratowski-subdivision.*

Using this crossing shadow, we can define Kuratowski-constraints as

$$\sum_{\{e,f\}\in \mathrm{CrPairs}(K)} x_{\{e,f\}} \geq 1 - \sum_{a\in\mathcal{X}_K[\bar{x},\bar{y}]} (1-x_a) - \sum_{b\in\mathcal{Y}_K[\bar{x},\bar{y}]} (1-y_b) \qquad (6)$$

for all LO-feasible integer vectors $(\bar{x},\bar{y})$ and all Kuratowski-subdivisions K in $G[\bar{x},\bar{y}]$. Here and in the sequel, $\mathrm{CrPairs}(K)$ denotes all pairs of edges belonging to different paths p_1, p_2 in K which may cross in order to planarize K (i.e., the edges corresponding to p_1 and p_2 in the underlying K_5 or $K_{3,3}$ are non-adjacent). Our constraints require at least one crossing on every Kuratowski-subdivision if it exists; this existence is detected via the crossing shadow.

Lemma 1. *Each optimal solution to the crossing number problem of any graph G corresponds to a feasible integer solution vector.*

Proof. Clearly, any solution to the crossing number problem can be described by an integer LO-feasible solution $(\bar{x},\bar{y})$ by construction, see Proposition 1. We show that this vector does not violate any constraint (6). Assume there is some $(\bar{x},\bar{y})$ and K which induces a violated Kuratowski constraint. Then

$$\sum_{\{e,f\}\in \mathrm{CrPairs}(K)} x_{\{e,f\}} < 1 - \sum_{a\in\mathcal{X}_K(\bar{x},\bar{y})} (1-x_a) - \sum_{b\in\mathcal{Y}_K[\bar{x},\bar{y}]} (1-y_b)$$

Since we only consider integer solutions, the left-hand side is 0 while the right-hand side is 1. We thus have:

$$\forall\{e,f\} \in \mathrm{CrPairs}(K) : x_{\{e,f\}} = 0 \text{ , and} \qquad (7)$$

$$\forall a \in \mathcal{X}_K[\bar{x},\bar{y}] : x_a = 1 \ \wedge \ \forall b \in \mathcal{Y}_K(\bar{x},\bar{y}) : y_b = 1\,.$$

But then, due to Proposition 2, the crossing shadow of $(\bar{x},\bar{y})$ w.r.t. K specifies exactly the crossings which induce a graph $\bar{G}$ that contains K as a Kuratowski-subdivision. Due to (7) we know that there are no further crossings on K which would lead to a planarization of this non-planar subgraph. This is a contradiction to the feasibility of the original solution. □

Lemma 2. *Every feasible solution to the ILP*

$$\min \left\{ \sum_{\{e,f\} \in \binom{E}{2}} c_e c_f x_{\{e,f\}} \text{ subject to (2),(3),(4),(5) and all (6)} \right\}$$

corresponds to a feasible solution of the crossing number problem.

Proof. We can interpret any integer LO-feasible solution $(\bar{x}, \bar{y})$ as a (partial) planarization $\bar{G} := G[\bar{x}, \bar{y}]$. Assume the solution vector satisfies all Kuratowski constraints, but $\bar{G}$ is non-planar. Then there exists a Kuratowski-subdivision in $\bar{G}$. Let K be such a subdivision with the smallest number of contained dummy nodes. We construct a crossing shadow $(\mathcal{X}_K[\bar{x}, \bar{y}], \mathcal{Y}_K[\bar{x}, \bar{y}])$ which describes the precise crossing configuration necessary to identify K. Since K is a non-planar (minimal) Kuratowski-subdivision, we know that there are no crossings on any pair of CrPairs(K). But then, (6) is violated for K and $(\mathcal{X}_K[\bar{x}, \bar{y}], \mathcal{Y}_K(\bar{x}, \bar{y}))$, as the left-hand side sums up to 0 and the right-hand side is 1. □

We therefore obtain:

Theorem 1. *Every optimal solution of the above ILP yields an optimal solution of the crossing number problem.*

3 Branch-and-Cut-and-Price Algorithm

The presented ILP

$$\min \left\{ \sum_{\{e,f\} \in \binom{E}{2}} c_e c_f x_{\{e,f\}} \text{ subject to (2),(3),(4),(5) and all (6)} \right\}$$

can be solved by a Branch-and-Cut framework: we start the computation with a subset of the above constraints and solve the LP-relaxations, i.e., we ignore the integer properties of the variables. Based on the thereby obtained fractional solution we start a *separation* routine to identify violated constraints not included in the current model. If we can find any, we add them to our model and iterate the process, computing the LP relaxation of this, now larger, model. If we cannot identify any more violated constraints but the solution is still not integer feasible, we have to resort to branching: we generate two subproblems, e.g., by fixing a variable to 0 and 1, respectively. Using the LP relaxations for lower bounds and some constructive heuristics for upper bounds, we can prune irrelevant subproblems.

Consider any optimal solution for any graph: at least half of the y-variables will be zero. Most graphs occurring in practice are far from being complete, and so actually most of the ILP variables will be zero in the optimal solution. Hence we augment the Branch-and-Cut framework with a column generation scheme, i.e., we start only with a subset of variables and assume that all other variables are zero. The task of the scheme is to detect which variables are necessary to add to the model, in order to guarantee overall optimality of the solution.

3.1 Upper Bounds and Integer Interpretation

To obtain upper bounds for our problem, we use the efficient planarization heuristic described in [11, 12]. As the experiments in [8] showed, this heuristic is very good in practice, often finding the optimal solution. Before the actual ILP computation is started, we use the heuristic to obtain a first upper bound.

During the computation, we compute LO-feasible *integer interpretations* $(\tilde{x}, \tilde{y})$ of the current fractional solution $(\bar{x}, \bar{y})$. We can then construct $G[\tilde{x}, \tilde{y}]$ and solve the crossing number problem heuristically on this partial planarization. The union of the crossings in $(\tilde{x}, \tilde{y})$ and the heuristic solution on $G[\tilde{x}, \tilde{y}]$ then constitutes a heuristic solution for G.

Since we require the integer solution $(\tilde{x}, \tilde{y})$ to be LO-feasible in order to construct the planarization $G[\tilde{x}, \tilde{y}]$, we cannot use a simple rounding scheme on the y-variables. Our integer interpretation works as follows:

$\tilde{x}$-variables: We apply a traditional rounding scheme to $\bar{x}$. The variable $\tilde{x}_{\{e,f\}}$ is 1 iff $\bar{x}_{\{e,f\}} > \tau$. Here $\tau > 0.5$ is a fixed threshold value; in our experiments we used $\tau = 0.7$ and $\tau = 1 - \epsilon$ (for some very small $\epsilon > 0$) and compute two probably distinct planarizations for the subsequent steps.

$\tilde{y}$-variables: Based on $\tilde{x}$, we can then restrict the set of $\tilde{y}$-variables that may be 1. For each edge e, let $\mathcal{D}_e$ be the set of edges which cross e, according to $\tilde{x}$. We can set $\tilde{y}_{e,f,g} = 0$ for all variables with $\{f, g\} \not\subseteq \mathcal{D}_e$. If $|\mathcal{D}_e| \geq 2$, we define a complete bidirected weighted graph, using $\mathcal{D}_e$ as its vertex set. We choose the weight of an arc (f, g) as $\bar{y}_{e,f,g}$. Then we solve the linear ordering problem on this graph, using a straight-forward greedy heuristic [1]. Using this resulting order, we can decide the values for $\tilde{y}_{e,f,g}$, for all $\{f, g\} \subseteq \mathcal{D}_e$.

3.2 Initial Constraints and Separation

We start our ILP only with the 0/1 bounds on the x-variables. Initially, we do not need to add the LO-constraints (2),(3),(4), and (5) for the y-variables, as these variables do not enter the objective function, cf. Section 3.3. All required Kuratowski-constraints (6) will be added during our cutting step.

There is no known efficient method to identify violated Kuratowski-constraints in a fractional solution, hence we only separate heuristically. We re-use the integer interpretation of fractional solutions as described in the previous section, and run a linear planarity test on $G[\tilde{x}, \tilde{y}]$. State-of-the-art planarity testing algorithms can efficiently (i.e., in linear time) extract a Kuratowski-subdivision as a certificate for non-planarity. We use the method presented in [10], which is a significantly modified variant of the planarity testing algorithm of Boyer and Myrvold [3], to efficiently extract several such certificates in linear time. For each obtained Kuratowski-subdivision, we then can compute the corresponding crossing shadow and test whether the resulting Kuratowski-constraint is violated, adding it to the LP if necessary.

3.3 Combinatorial Column Generation

Our initial linear program only contains the x-variables. Note that only these variables enter the objective function: the values of the y-variables do not influence the solution value as they are only introduced to solve the ordering problems on the edges. Furthermore, we do not require y-variables if there is only a single crossing on all edges – then all y-variables are zero. Hence, conceptually, having some solution $\bar{x}$, we only require the y-variables with a base edge e, if there are multiple edges crossing over e. Since the separation routine does only use integer interpretations of the current solution, we only require the knowledge of the crossing order if $\sum_{f \in E \setminus \{e\}} \tilde{x}_{\{e,f\}} \geq 2$. Let F_e be the set of edges f with $\tilde{x}_{\{e,f\}} = 1$. The order of performing the variable generation prior to the separation routine is critical: we first obtain a fractional solution and check if the solution can be uniquely interpreted as a partial planarization, i.e., if all the variables $y_{e,f,g}$, with $\{f, g\} \subseteq F_e$, are contained in the current LP model. If there is at least one such y-variable missing in the current LP model, we add all required such variables, together with their corresponding LO-constraints, and resolve our LP model.

Hence, the variable generation takes place before we interpret a fractional solution as a partial planarization for the separation routine, and before the bounding heuristic. Therefore, for these steps we guarantee that all necessary y-variables are in the model, and the solution is LO-feasible.

3.4 Branching on K_5-constraints

We can use Kleitman's parity argument for complete graphs with an odd number of vertices [13, 14]: if a K_{2n+3}, $n \in \mathbb{N}^+$, has an even or odd crossing number, every possible drawing of K_{2n+3} will also have an even or odd number of crossings, respectively. Since we know that $\mathrm{cr}(K_5) = 1$, we have for every K_5-subdivision that if it is drawn with more than one crossing, it will require at least 3 crossings.

This jump in the crossing number can be used for branching. Most commonly, we would select a variable z and generate two subproblems with $z = 0$ and $z = 1$. Before we resort to this kind of branching, we check for any K_5-constraint of the type $p^T x + q^T y \geq 1$, with p and q being the coefficient vectors. We can then generate two subproblems, one with $p^T x + q^T y = 1$ and one with $p^T x + q^T y \geq 3$. Note that, theoretically, we can continue to branch on the latter constraint, generating $p^T x + q^T y = 3$ and $p^T x + q^T y \geq 5$, etc.

4 Further Remarks

Extending OOCM**.** The SOCM ILP was extended, e.g., to compute the bimodal crossing number [5], the minor-monotone and hypergraph crossing numbers [6], and the simultaneous crossing number [9]. The extensions for the first three problems can straight-forwardly be formulated within OOCM.

By contrast, extensions for the simultaneous crossing number, as well as potential extensions for the pairwise and the odd crossing number [18] are not

straight-forward: they require that some edges cross multiple times, maybe even an exponential number of times. This states no problem for SOCM, as we can, theoretically, subdivide the edges into long enough paths and drop the one-crossing-per-edge-pair constraint. Anyhow, we cannot model such multiple crossings with the variables of OOCM.

Improvements to SOCM. The SOCM implementation of our experiments received improvements compared to the algorithm presented in [4, 8]. Hence, the results are far better than previously reported. We denote the improved version of SOCM by iSOCM. The modifications include:
– The crossing minimization heuristic (used by both iSOCM and OOCM) improved, due to a more time-consuming but stronger post-processing scheme: in [11], the strongest post-processing was to remove and reinsert every edge, after obtaining a first full solution. The current implementation in OGDF [17] can remove and reinsert all edges after each single edge-insertion step.
– The branching on K_5-constraints, cf. Section 3.4, is also possible in iSOCM.
– The column generation scheme is now fine-tuned: originally, we introduced a new segment of the original edge e whenever the sum of crossings over the first segment of e is larger than 1 in the fractional solution. Now, we add this segment only if the sum is larger than 1 in the rounded solution that is used for the separation. This idea is then similar to the generation criterion in OOCM.
– As OOCM, iSOCM also uses the new extraction algorithm which finds multiple Kuratowski-subdivisions in linear time [10].

5 Experiments

The following experiments were conducted on an AMD Opteron 2.4 GHz with 2GB of RAM per process. SOCM, iSOCM, and OOCM are implemented in the open-source library OGDF [17], using ABACUS as a B&C framework and CPLEX 9.0 as LP solver. We applied a time limit of 30 minutes for each instance. The machine and the overall experimental setting is thus identical to the experiments reported in [4, 8], which yielded the currently best known published results.

To compare the performance of both formulations, we chose the well-known *Rome* benchmark set [2], which is commonly used to assess algorithms for the crossing number and other graph drawing problems, e.g. [4, 8, 11]. It consists of over 11,500 real-world graphs emerging from software-engineering applications, with between 10 and 100 nodes. We use the non-planar core reduction [7] as a preprocessing step. We say graphs are *trivial*, if they are planar or if the heuristic achieves a planarization with only one crossing, as in these cases we need not prove optimality. The Rome library contains 7172 non-trivial graphs.

As we see in Figure 1, both new algorithms clearly outperform the old SOCM algorithm, which drops below a success-ratio of 50% for graphs with 70 nodes. While OOCM solves virtually all graphs with up to 60 nodes to provable optimality within the time limit, the formerly best algorithm already drops to a 70% success-ratio for graphs of size 60. The experiments also show that the new ILP

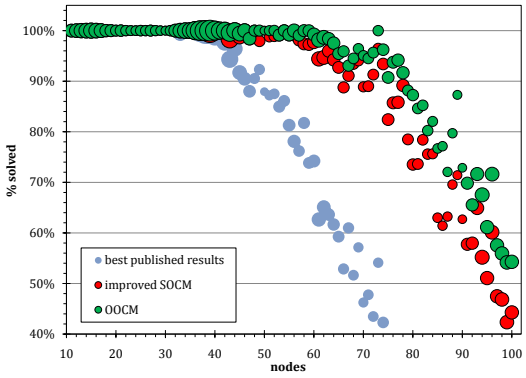

Fig. 1. Percentage of graphs solved to provable optimality within 30 minutes. The size of the circles denotes the number of instances per graph size. Therefore, larger circles correspond to statistically more reliable data points. The gray data points denote the previously best published results in [8] and the journal version of [4].

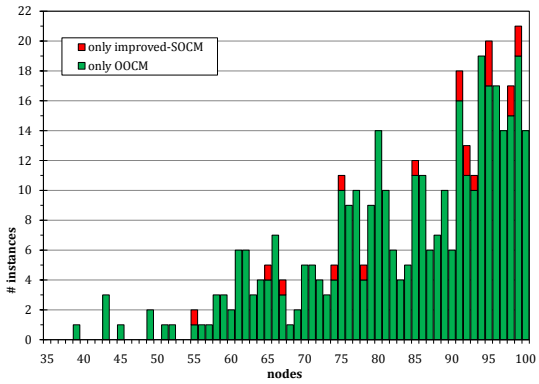

Fig. 2. The number of instances only solved by one of the approaches, but not by both.

formulation OOCM is able to solve more and larger graphs than iSOCM: while iSOCM only solves 84.4% of all non-trivial graphs within 30 minutes, OOCM finds and proves an optimal solution in 89.2% of all these instances, i.e., 93.3% over all benchmark instances. Even when OOCM has a time limit of only 10 and 5 minutes per non-trivial instance, it still solves 85.9% and 83.4%, respectively, and thus produces results comparable to 30 minutes of iSOCM computation in a 3–6x shorter period of time.

Note that there are only 19 instances solved by iSOCM but not by OOCM, within 30 minutes, but 361 instances which OOCM solved but iSOCM did not, cf. Figure 2. Most importantly, we can now solve over 50% of the largest graphs of the Rome library. Figure 3 further illustrates the strength of OOCM; it shows the average running times for graphs solved by both approaches; even for large graphs OOCM only requires roughly 100 seconds on average.

Figure 4 shows the dependency of the solvability on the crossing number: we see that OOCM solves all but 6 graphs with a crossing number of up to 20. It even solves a graph with a crossing number of 37. By contrast, iSOCM solves only all but 7 graphs with a crossing number of at most 12. Finally, Figure 5 shows a comparison of the number of required variables for the instances solved by both approaches: both algorithms start with the same initial variable set,

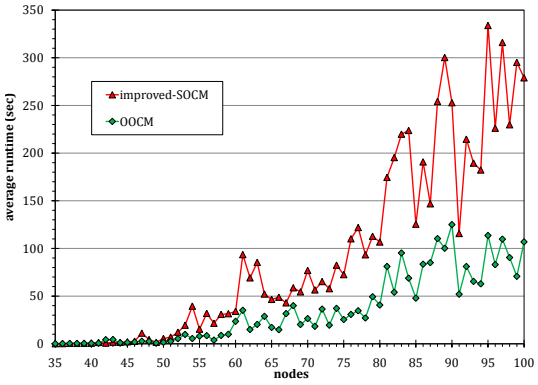

Fig. 3. The required running time, averaged over the instances solved by both OOCM and SOCM.

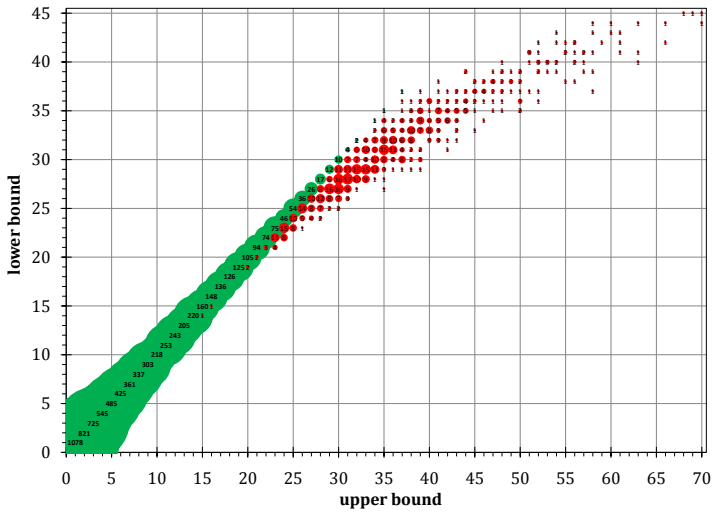

Fig. 4. The number of instances per combination of lower and upper bound after 30 minutes of OOCM, over all graphs of the Rome library. 9 instances are not shown as their lower or upper bounds do not fit into this diagram.

but OOCM requires by far less additional variables during the computation of the optimal solution. This seems to be the main reason why OOCM is faster and more efficient than SOCM and iSOCM.

References

1. G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis. *Graph Drawing: algorithms for the visualization of graphs*. Prentice Hall, 1999.
2. G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An experimental comparison of four graph drawing algorithms. *Comput. Geom. Theory Appl.*, 7(5-6):303–325, 1997.
3. J. M. Boyer and W. J. Myrvold. On the cutting edge: Simplified $O(n)$ planarity by edge addition. *Journal of Graph Algorithms an Applications*, 8(3):241–273, 2004.
4. C. Buchheim, M. Chimani, D. Ebner, C. Gutwenger, M. Jünger, G. W. Klau, P. Mutzel, and R Weiskircher. A branch-and-cut approach to the crossing number problem. *Discrete Optimization*, 5:373–388, 2008. In Memory of George B. Dantzig.

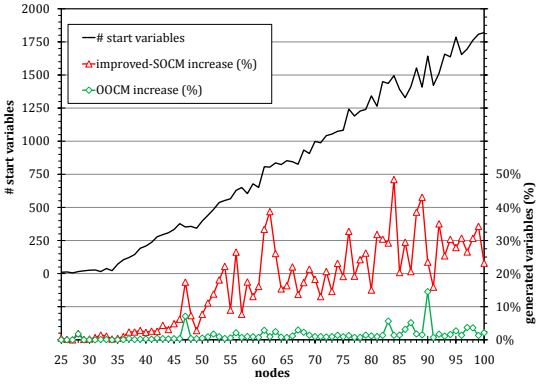

Fig. 5. The average factor by which the number of variables increases compared to the number of start variables, which is identical for OOCM and SOCM. The diagram also shows the average number of start variables per graph size.

5. C. Buchheim, M. Jünger, A. Menze, and M. Percan. Bimodal crossing minimization. In *Proc. COCOON '06*, volume 4112 of *LNCS*, pages 497–506, 2006.
6. M. Chimani and C. Gutwenger. Algorithms for the hypergraph and the minor crossing number problems. In *Proc. ISAAC '07*, volume 4835 of *LNCS*, pages 184–195, 2007.
7. M. Chimani and C. Gutwenger. Non-planar core reduction of graphs. *Discrete Mathematics*, 2007. to appear. A preliminary version appeared in *Proc. GD '05*, LNCS 3843, pp. 223–234.
8. M. Chimani, C. Gutwenger, and P. Mutzel. Experiments on exact crossing minimization using column generation. In *Proc. WEA '06*, volume 4007 of *LNCS*, pages 303–315, 2006.
9. M. Chimani, M. Jünger, and M. Schulz. Crossing minimization meets simultaneous drawing. In *Proc. IEEE PacificVis '08*, 2008.
10. M. Chimani, P. Mutzel, and J.M. Schmidt. Efficient extraction of multiple kuratowski subdivisions. In *Proc. GD '07*, volume 4875 of *LNCS*, pages 159–170, 2008.
11. C. Gutwenger and P. Mutzel. An experimental study of crossing minimization heuristics. In *Proc. GD '03*, volume 2912 of *LNCS*, pages 13–24, 2004.
12. C. Gutwenger, P. Mutzel, and R. Weiskircher. Inserting an edge into a planar graph. *Algorithmica*, 41(4):289–308, 2005.
13. D.J. Kleitman. The crossing number of $K_{5,n}$. *J. Comb. Theory*, 9:315–323, 1970.
14. D.J. Kleitman. A note on the parity of the number of crossings of a graph. *J. Comb. Theory, Ser. B*, 21(1):88–89, 1976.
15. J. Kratochvíl. String graphs II: Recognizing string graphs is NP-hard. *J. Combin. Theory Ser. B*, 52:67–78, 1991.
16. C. Kuratowski. Sur le problème des courbes gauches en topologie. *Fund. Math.*, 15:271–283, 1930.
17. *OGDF – Open Graph Drawing Framework*. See `http://www.ogdf.net`, 2008.
18. J. Pach and G. Tóth. Which crossing number is it anyway? *J. Comb. Theory Ser. B*, 80(2):225–246, 2000.
19. M. Schaefer, E. Sedgwick, and D. Štefankovič. Recognizing string graphs in NP. *Journal of Computer and System Sciences*, 67(2):365–380, 2003.
20. I. Vrt'o. Crossing numbers of graphs: A bibliography. See `ftp://ftp.ifi.savba.sk/pub/imrich/crobib.pdf`, 2007.