Algorithms for the Hypergraph and the Minor Crossing Number Problems

Markus Chimani and Carsten Gutwenger

Department of Computer Science, University of Dortmund, Germany
{markus.chimani,carsten.gutwenger}@cs.uni-dortmund.de

Abstract. We consider the problems of hypergraph and minor crossing minimization, and point out a relation between these two problems which has not been exploited before. We present some complexity results regarding the corresponding edge and node insertion problems. Based on these results, we give the first embedding-based heuristics to tackle both problems and present a short experimental study. Furthermore, we give the first exact ILP formulation for both problems.

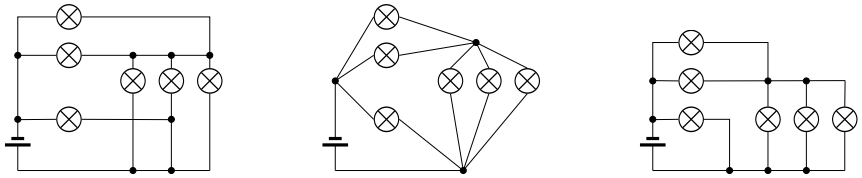
1 Introduction

A *drawing* of a graph G on 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. The *crossing number* $\mathrm{cr}(G)$ then is the smallest number of crossings in any drawing of G. The corresponding crossing minimization (CM) problem has been widely studied in the literature, see [15] for an extensive bibliography.

In this paper we consider the problem of finding the *Hypergraph Crossing Number*, as defined in the following section. We do so by exploiting a connection between this crossing number and the *Minor Crossing Number*, i.e., the smallest crossing number of any graph G' which has G as its minor, denoted by $G \preceq G'$. Especially the latter concept has yet only been studied in the context of theoretical lower and upper bounds [2, 3], but was never before tackled algorithmically.

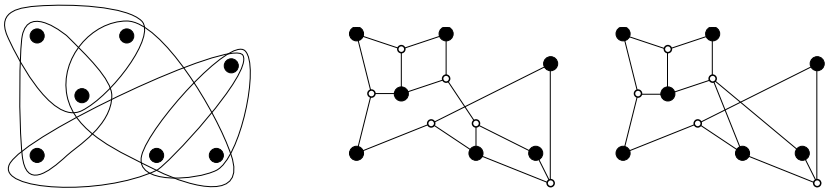
Besides from their theoretical appeal, these problems also occur, e.g., for crossing minimal layouts of electrical wiring schemes [3], cf. Fig. 1. Usually, the exact topology of such a wiring scheme G'' is not interesting for the connected subgraphs which have the same electric potential. Hence we can "merge" these nodes into one node, which is exactly the operation to obtain a minor G, compute the minor crossing number $\mathrm{mcr}(G)$ and expand the graph accordingly to obtain G'. In this example, we can observe the connection to hypergraphs: by seeing the impedances on the wires as nodes, we can interpret the wires on the same potential as *hyperedges*, i.e., edges with multiple incident nodes.

The computation of both crossing numbers is NP-complete. In this context, we investigate several subproblems of edge and node insertion (Sections 3 and 5),



(a) (b) (c)

Fig. 1. The wiring scheme (a) cannot be drawn without any crossing. By computing a minor (b) and considering a realizing graph (c), we obtain an equivalent but planar wiring scheme.



(a) Subset (b) Edge, tree-based (c) Edge, point-based

Fig. 2. A hypergraph, drawn using different drawing styles.

and establish novel heuristics for both kinds of crossing numbers (Section 4). We also implemented these algorithms and give an experimental study in Section 7. Furthermore, in Section 6 we sketch the first exact ILP formulation able to provably solve both problems to optimality.

The next subsections will give the detailed definitions of the considered problems and summarize our results.

1.1 Definitions

A hypergraph $H = (V, \mathcal{F})$ differs from an ordinary graph that instead of edges—which have exactly two incident nodes—we consider *hyperedges*: a hyperedge $F \in \mathcal{F}$ is a proper subset of V, i.e., $F \subset V$, with $|F| \geq 2$. See, e.g., [10] for details. There are two major variants on how to draw hypergraphs [12], cf. Fig. 2: the *subset-standard* and the *edge-standard*. The first variant becomes very confusing with more hyperedges, and it is ambiguous how to define a consistent notion of crossings. Hence, most applications, cf. [5, 11, 14], focus on the edge-standard which allows two subvariants: in the *tree-based* drawing style, each hyperedge F is drawn as a tree-like structure of lines, whose leaves are the incident nodes of F. If we restrict the tree-like structure of every hyperedge to be a star, we obtain the *point-based* drawing style.

Formally, each hyperedge $F \in \mathcal{F}$ has a set of associated *hypernodes* N_F, which form the branching points of the line tree: each node $v \in F$ is connected to exactly one $n \in N_F$, and all hypernodes N_F are treewise connected. By this *tree-based transformation* we obtain a traditional graph L. We denote the set

of all graphs L obtainable by such transformations by $\mathcal{L}(H)$, and can naturally define:

Definition 1 (Tree-based Hypergraph Crossing Number). Let H be a hypergraph. We define the *tree-based hypergraph crossing number* as

$$\text{thcr}(H) := \min_{L \in \mathcal{L}(H)} \text{cr}(L).$$

The tree-based hypergraph crossing number has the elegant property that it is equivalent to the traditional crossing number if all hyperedges have cardinality 2. Because of this property, computing $\text{thcr}(H)$ is NP-complete.

We further define the *point-based transformation* $\Lambda(H)$ as the special tree-based transformation, where each hyperedge has exactly one associated hypernode, i.e., $\Lambda(H) := (V \cup \mathcal{F}, E(H))$ with $E(H) := \{(v, F) \mid v \in F, F \in \mathcal{F}\}$. Clearly, this leads to the point-based drawing style and the definition of the *point-based hypergraph crossing number* $\text{phcr}(H) := \text{cr}(\Lambda(H))$.

Observation 1. For any $L \in \mathcal{L}(H)$ we have $\Lambda(H) \preceq L$, i.e., the point-based transformation of H is the minor of any tree-based transformation of H.

Point-based hypergraph planarity of H can be defined as $\text{phcr}(H) = 0$ straightforwardly and is equivalent to Zykov planarity [10]. It can be efficiently tested by transforming H into $\Lambda(H)$ in linear time and applying any traditional linear-time planarity testing algorithm to $\Lambda(H)$. Analogously, *tree-based hypergraph planarity* can be defined as $\text{thcr}(H) = 0$. Since $L \in \mathcal{L}(H)$ is planar if and only if $\Lambda(H)$ is planar, all three planarity definitions are equivalent.

Obviously, the point-based hypergraph crossing minimization of H is equivalent to the traditional crossing minimization on the graph $\Lambda(H)$. Hence we will focus on computing $\text{thcr}(H)$. To understand this crossing number better, we first have to focus on the *minor crossing number* [3], also known as the *minor-monotone crossing number*, for traditional graphs:

Definition 2 (Minor Crossing Minimization Problem). Let $G = (V, E)$ be an undirected graph. The *Minor Crossing Minimization Problem* (MCM) is to find a *realizing graph* $G' = (V', E') \succeq G$ with $\text{cr}(G') = \text{mcr}(G) := \min_{G'' \succeq G} cr(G'')$, i.e., the minor crossing number of G.

Let $W \subseteq V$. We can define a minor relation $G' \succeq_W G$, i.e., G is a W-minor of G' if we can obtain G' from G by only expanding nodes of W. This leads to the more general *W-restricted Minor Crossing Number* $\text{mcr}_W(G)$, i.e., the smallest crossing number of any graph $G' \succeq_W G$, and the *W-restricted Minor Crossing Minimization Problem* (RMCM). Clearly $\text{mcr}_W(G) = \text{mcr}(G)$, if $W = V$. Since nodes with degree less than 4 are irrelevant for the differences between the traditional and the minor crossing number, we have:

Proposition 1. *Let* $\hat{\mathcal{F}} := \{F \in \mathcal{F} \mid |F| \geq 4\}$. *The tree-based hypergraph crossing number is equivalent to the* $\hat{\mathcal{F}}$*-restricted minor crossing number of* $\Lambda(H)$*, i.e.,* $\text{thcr}(H) = \text{mcr}_{\hat{\mathcal{F}}}(\Lambda(H))$.

Hence we have to find a realizing graph $\Lambda' \succeq_{\mathcal{F}'} \Lambda(H)$ with smallest crossing number, i.e., we may obtain Λ' only by expanding hypernodes of degree at least 4.

1.2 Edge and Node Insertion Results

The MCM problem was first stated in [3] and was shown to be NP-complete in [9]. In Section 2, we will discuss two alternative viewpoints of MCM, which are fundamental for the results presented thereafter. In the following we will always consider the W-restricted variant of the minor crossing number. Since W may be the full set V, the results clearly also hold for the traditional minor-monotonous case. Note that we consider two embeddings, or rotation-systems, Γ of G and Γ' of G' ($G \preceq G'$) *equivalent*, if we can obtain Γ by performing the necessary minor operations stepwise on G' and Γ' in the natural way: let us merge the connected nodes a and b with their respective cyclic orders $\pi_a = \langle \{a, b\}, e_1, \ldots, e_{\deg(a)-1} \rangle$ and $\pi_b = \langle \{b, a\}, f_1, \ldots, f_{\deg(b)-1} \rangle$ of their incident edges. The new node c will have the cyclic order $\pi_c = \langle e_1, \ldots, e_{\deg(a)-1}, f_1, \ldots, f_{\deg(b)-1} \rangle$.

Similar to the corresponding problems for the traditional crossing number, we can state the following related problems:

Definition 3 (Minor Edge Insertion). Let $G = (V, E)$ be a planar undirected graph, and let $e = \{s, t\} \in V \times V \setminus E$ be an edge not yet in G. The *W-restricted Minor Edge Insertion Problem with Variable Embedding (MEIV)* is to find the W-restricted minor crossing number of the graph $G + e$, under the restriction that the realizing drawing induces a planar drawing of G.

Given a specific planar embedding Γ of G, the *W-restricted Minor Edge Insertion Problem with Fixed Embedding (MEIF)* is to find the W-restricted minor crossing number of the graph $G+e$, under the restriction that the realizing drawing induces an embedding of G equivalent to Γ.

For both problems, the equivalent problems concerning the traditional crossing number can be solved in linear time. In Section 3, we show:

Theorem 1. *MEIF and MEIV can be solved optimally in linear time.*

In Section 4, we show how to use this result to obtain a heuristic following the planarization approach.

Definition 4 (Minor Node Insertion). Let $G = (V, E)$ be a planar undirected graph, let $v \notin V$ be a node not yet in G, and let E' be edges connecting v with nodes of V. Let $W^- := W$ and $W^+ := W \cup \{v\}$. The *W^--restricted (W^+-restricted) Minor Node Insertion Problem with Variable Embedding* is to find the W^--restricted (W^+-restricted) minor crossing number of the graph $G' = (V \cup \{v\}, E \cup E')$, under the restriction that the realizing drawing induces a planar drawing of G. We abbreviate these problems MNIV$^-$ and MNIV$^+$, respectively.

Given a specific planar embedding Γ of G, the *W^--restricted (W^+-restricted) Minor Node Insertion Problem with Fixed Embedding* is to find the W^--restricted (W^+-restricted) minor crossing number of the graph G', under the restriction that the realizing drawing induces an embedding of G equivalent to Γ. We abbreviate these problems MNIF$^-$ and MNIF$^+$, respectively.

The equivalent problem for the traditional crossing number and a fixed embedding can be solved in $\mathcal{O}(|V| \cdot |E'|)$ time. An analogous algorithm, together with the ideas of Theorem 1, can be used to show (see Section 5):

Theorem 2. *MNIF$^-$ is solvable in $\mathcal{O}(|V| \cdot |E'|)$ time.*

The problem for the traditional crossing number where all embeddings are considered, and therefore a special case of MNIV$^-$, is still an open problem. In contrast to these results, we can show that the problem is hard when the inserted node is allowed to be expanded:

Theorem 3. *MNIF$^+$ and MNIV$^+$ are NP-complete. This also holds for the case $W = V$, i.e., non-restricted minor-monotonicity.*

Corollary 1. *Let $H = (V, \mathcal{F})$ be a hypergraph, and $F \notin \mathcal{F}$ a hyperedge not yet in H. Under the restriction that H has to be drawn planar and independent on whether a specific embedding of H is given or not, we have: computing* $\mathrm{thcr}(H + F)$ *is NP-complete.*

The theorem is based on the observation that we can turn any planar Steiner tree problem instance (NP-complete, [6]) into a corresponding MNIF$^+$ problem, cf. Section 5. The corollary does not follow from Theorem 3 itself, but from its proof. Furthermore, since computing $\mathrm{thcr}(H)$ is a special case of a W-resticted minor crossing number we have in particular:

Observation 2. The heuristic and exact algorithms presented below can be used to solve the tree-based hypergraph crossing number problem heuristically and to optimality, respectively.

2 General Observations

In the following, we will always consider an undirected graph $G = (V, E)$ with $W \subseteq V$, and we are interested in $\mathrm{mcr}_W(G)$. For the following algorithms, there are two points of view which are helpful when discussing the problem of minor crossing numbers: we can replace each node $v \in W$ with $\deg(v) \geq 4$ by an *expansion tree* T_v, which is incident to all nodes—or their respective expansion trees—originally incident to v. The nodes of T_v are called the *split nodes* of v. The RMCM problem can then be reformulated as finding a *tree expansion* G', i.e., a graph obtained by such transformations, with smallest crossing number.

Another possibility to view the problem is that in the traditional crossing number problem, edges are allowed to cross. For the minor crossing number, edges are allowed to cross through vertices, and moreover vertices may even "cross" other vertices. Such crossings can be seen as crossings between an expansion tree and a traditional edge, or between two expansion trees, respectively.

3 Optimal Edge Insertion

In this Section, we present linear time algorithms for MEIF and MEIV. Our task is to find a tree expansion G' of G along with an insertion path connecting s and t, i.e., an ordered list of edges that are crossed when inserting e. Observe that it is never necessary to expand s or t.

Fixed Embedding. Let Γ be an embedding of G. We define a directed graph $D_{\Gamma,s,t} = (N, A)$ as follows. N contains a node n_f for each face $f \in \Gamma$ and a node n_v for each node $v \in W \cup \{s, t\}$. Each arc $a \in A$ has an associated cost $c_a \in \{0, 1\}$; we have the following arcs:

- For each pair f, f' of adjacent faces, we have two arcs $(n_f, n_{f'})$ and $(n_{f'}, n_f)$ with cost 1.
- For each node $v \in W \setminus \{s, t\}$ and face f incident to v we have an arc (n_v, n_f) with cost 1 and an arc (n_f, n_v) with cost 0.
- Finally, we have arcs (n_s, n_f) for each face f incident to s and (n_f, n_t) for each face f incident to t; all these arcs have cost 0.

Then, the solution to MEIF is the length of a shortest path p in $D_{\Gamma,s,t}$ from n_s to n_t; each arc $(n_f, n_{f'})$ in p corresponds to crossing an edge separating f and f', and each subpath $(n_f, n_v), (n_v, n_{f'})$ corresponds to splitting node v and crossing the edge resulting from the split.

The number of nodes in N is bounded by $|V| + |F|$, where F is the set of faces in Γ, and the number of arcs in A by $4 \cdot |E|$, since we have at most four arcs per edge. Hence, we can apply breadth-first-search for finding a shortest path in $D_{\Gamma,s,t}$ which takes time $\mathcal{O}(|V|+|E|)$. We remark that BFS can easily be extended to graphs with 0/1-arc costs. Thus, we can solve MEIF in linear time.

Variable Embedding. In order to solve MEIV, we adapt the algorithm by Gutwenger et al. [8] which solves the problem for the traditional crossing number, i.e., $W = \emptyset$ and no tree expansions are possible. They showed that it is sufficient to consider the shortest path $B_0, v_1, B_1, \ldots, v_k, B_k$ in the BC-tree of G and independently compute optimal edge insertion paths in the (biconnected) blocks B_i from v_i to v_{i+1} ($0 \leq i \leq k$, $v_0 = s$, and $v_{k+1} = t$). This is also true when we are allowed to split the nodes W: concatenating the respective paths in the blocks results in a valid insertion path, and alternately crossing edges from different blocks or splitting (and crossing through) a cut vertex v_i would result in unnecessary crossings.

Thus, we can restrict ourselves to a biconnected graph G. Let $\mathcal{T}$ be the *SPQR-tree* of G (an SPQR-tree represents the decomposition of a biconnected graph into its triconnected components, please refer to [8] for details). We consider the shortest path $p = \mu_1, \ldots, \mu_h$ in $\mathcal{T}$ from a node μ_1 whose skeleton contains s to a node μ_h whose skeleton contains t. Let S_i be the skeleton of μ_i ($1 \leq i \leq h$). The *representative* $\text{rep}(v)$ of a node $v \in G$ in a skeleton S_i is either v itself if $v \in S_i$, or the edge $e \in S_i$ whose expansion graph contains v. If $W = \emptyset$, the optimal

algorithm only considers the R-nodes—triconnected components with therefore unique embeddings—on p and independently computes optimal edge insertion paths in fixed embeddings of the respective skeletons from $\text{rep}(s)$ to $\text{rep}(t)$. If the representative is an edge, we assume that a virtual node is placed on this edge and serves as start or endpoint of the insertion path.

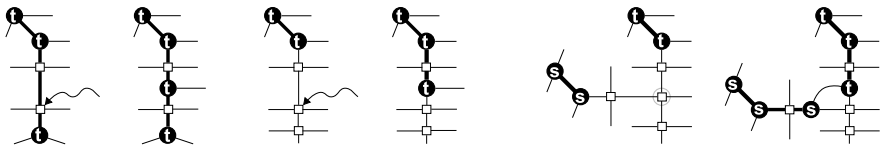
This approach is invalid if $W \neq \emptyset$: an optimal insertion path in a skeleton S_i might cross through an endpoint x of the edge representing t in S_i, and continuing this path from x in S_{i+1} might save a crossing. We circumvent this problem by processing p in order from μ_1 to μ_h: for each R-node μ_i where $\text{rep}(t)$ is an edge $e_t = (x, y)$, we compute three insertion paths p_x, p_y, and p_e in a fixed embedding of S_i, which are optimal insertion paths to x, y, and e_t, respectively. Observe that for the respective lengths ℓ_x, ℓ_y, and ℓ_e of these paths we have $\ell_e \leq \ell_x, \ell_y \leq \ell_e + 1$. If $x \in W$, $\ell_x = \ell_e$, and x is contained in the skeleton of the next processed R-node μ_j, then x is a possible start node for an insertion path in S_j; the analogous is true for y; $\text{rep}(s)$ is always a possible start node. We compute the optimal insertion paths in the R-node skeletons by slightly modifying the search network introduced for MEIF. We introduce a super start node s^* and connect it to the possible start nodes. Then we compute shortest paths from s^* to $\text{rep}(t)$ and, if this is an edge e_t, to the endpoints of e_t.

After processing all nodes on p, we reconstruct the optimal insertion path backwards from t to s. The insertion path in S_h ends in t; we determine which insertion path in the preceding R-node skeleton to chose by checking which start node is used, until we reach s. This algorithm can be implemented in linear time, thus showing that MEIV can be solved in linear time as well.

4 The Planarization Approach for RMCM

The planarization approach is a well-known and successful heuristic for traditional crossing minimization; see [7] for an experimental study. First, a planar subgraph is computed, and then the remaining edges are inserted one after another by computing edge insertion paths and inserting the edges accordingly, i.e., edge crossings are replaced by dummy vertices with degree 4.

In order to apply the algorithms from the previous section in a planarization approach for RMCM, we need to generalize them, since the insertion of edges splits nodes and thereby expands them to trees. Furthermore, edges of G and edges resulting from node splits get subdivided by dummy vertices during the course of the planarization. We call the resulting paths *edge paths* and *tree paths*, respectively. Hence, we are not simply given two nodes s and t but two node sets S and T, and we have to find an insertion path connecting a node of S with a node of T. Thereby, S (T) is the set of all split nodes of s (t) and all dummy nodes on edge or tree paths starting at a split node of s (t). The dummy nodes in these sets have the property that a simple extension of a tree expansion is sufficient to connect an insertion path to a correct split node; see Fig. 3 for a visual description.



(a) Dummy on target (b) Dummy near target (c) Dummy near source and target

Fig. 3. Modification of insertion paths ending at dummy nodes. Bold solid edges are part of expansion trees, dummy-nodes are denoted by squares.

Before we discuss the details, we give an overview of the planarization approach for RMCM.

(1) Compute a planar subgraph $G' = (V, E_P)$ of G.
(2) For each edge $e = \{s, t\} \in E \setminus E_P$:
   (a) Compute S and T.
   (b) Find an insertion path p from S to T in G'.
   (c) Insert e into G' according to p by splitting nodes if required and introducing new dummy nodes for crossings.

It remains to show how to generalize the edge insertion algorithms. In the fixed embedding scenario, we simply introduce a super start node s^* connected to all nodes in S, and a super end node t^* connected from all nodes in T in the search network. The following lemma shows the key property for generalizing the variable embedding case.

Proposition 2. *1. The blocks of G' containing a node in S (T) and the cut vertices of G' contained in S (T) form a subtree of the BC-tree of G'.*
2. Let $\mathcal{T}$ be the SPQR-tree of a block of G'. Then, the nodes of $\mathcal{T}$ whose skeletons contain a node in S (T) form a subtree of $\mathcal{T}$.

This allows us to compute the shortest paths in the BC- and SPQR-trees in a similar way as described above. The only difference is that we consider blocks and skeletons containing any node in S (or T). The computation of insertion paths in R-node skeletons is generalized as for the fixed case if several nodes of S or T are contained.

In [7], two important improvement techniques for the planarization approach are described which are both also applicable for RMCR. The *permutation* strategy calls step (2) several times and processes the edges in $E \setminus E_P$ in random order. The *postprocessing* strategy successively removes an edge path and tries to find a better insertion path. This can also be done for tree paths which in fact is a key optimization of our approach, since it allows to introduce crossings between two tree expansions as well. Finally, we remark that we also contract tree paths during the algorithm if they no longer contain a dummy node and thus become redundant.

5 Optimal Node Insertion

In this section we prove the theorems regarding the minor node insertions.

Algorithm for Theorem 2. We show that MNIF$^-$ is solvable in $\mathcal{O}(|V| \cdot |E'|)$.

Let U be the nodes of V incident to edges of E'. We can solve the node insertion problem with fixed topology for the traditional crossing number by considering the dual graph D of G with respect to Γ. Each node in D is labeled with a number which is initially 0. We then start a BFS for each $u \in U$, augmenting D with edges between u and its incident faces. The nodes of U are incremented by the BFS-depth, for each different u. Finally, each node of D holds the sum of the shortest distances between itself and the nodes U. We then simply pick a node of D with smallest number, and insert the new node v into the corresponding face in Γ.

Using the ideas from the above sections, we can use the same algorithm but allowing edges to cross through nodes. Since all inserted edges are incident to v, they will not cross each other in any optimal node insertion. Therefore, no conflicting edge-node crossings can occur, other then ones based on paths with equal length. Such conflicts can easily be resolved by choosing any of the conflicting paths for both inserted edges. The correctness and running time of the algorithm follows directly. □

Proof of Theorem 3. We show that MNIF$^+$ and MNIV$^+$ are NP-complete.

We can restrict ourselves to MNIF$^+$. Since a planar 3-connected graph has only a unique planar embedding and its mirror, we naturally have NP-completeness for MNIV$^+$ if MNIF$^+$ is NP-complete. We only briefly sketch the idea of the proof.

We reduce to the planar Steiner tree problem, which is known to be NP-complete [6]. Thereby we are given a planar graph D with integer edge-weights and a subset of its nodes are marked as *terminals*. We ask for a weight-minimum tree T connecting all terminals (and possibly some other nodes). Let G be the dual of D, then finding the Steiner tree in D becomes equivalent to finding a treewise expansion for the node v which we want to insert into G minor-monotonously. Since the integer edge-weights can be bounded to be a polynomial in the graph size, we can replace each edge of weight w by a simple path of w edges, thus only polynomially enlarging the given graph. Hence we have NP-completeness for MNIF$^+$. □

6 An ILP formulation for RMCM

We briefly sketch how to construct an integer linear program to solve the RMCM, and therefore also the hypergraph crossing minimization problem, to optimality. We base our formulation on the ILP for CM, presented in [4]: its main idea is to have variables $x_{e,f}$ for each pair of edges e and f, which are 1 if the edges

cross and 0 otherwise. To circumvent problems with the realizability checking of solutions, the graph G is first modified such that each original edge is replaced by a simple path of multiple edges. The ILP is then based on *Kuratowski constraints*, i.e., for each K_5 and $K_{3,3}$ subdivision contained in (partial planarizations of) G we have an inequality which requires at least one crossing.

We can use this ILP by considering tree expansions of G. We replace each node $v \in W$, with $\deg(v) \geq 4$, by a vertex set V'_v with $|V'_v| = 2\deg(v) - 2$. Each edge originally incident to v is incident to a unique vertex of this set. By augmenting V'_v with edges, we can model any treewise connection of the vertices $\{w \in V'_v \mid \deg(w) = 1\}$. Hence, considering the edges E'_v of the complete graph on the set V'_v, we introduce 0/1-variables y_e for all $e \in E'_v$. If such a variable is 1, the corresponding edge is used for the treewise connection of V'_v.

We now require two main types of constraints: we can generalize the Kuratowski constraints straight-forwardly. Let R be the set of edges with y variables which are contained in some Kuratowski subdivision K. We can subtract the term $\sum_{e \in R}(1 - y_e)$ on the right-hand side of the corresponding $\geq$-constraint, i.e., we only require a crossing on K if all its edges are selected.

Additionally, we have to assure the treewise connection for each V'_v. Therefore we require to select exactly $|V'_v| - 1$ edges of E'_v for each set V'_v, and assure connectivity via traditional cut constraints:

$$\forall v \in W, \deg(v) \geq 4, \forall \emptyset \neq S \subset V'_v : \sum_{u \in S, w \in V'_v \setminus S} y_{\{u,w\}} \geq 1.$$

Note that all these constraints can be added dynamically within a Branch-and-Cut framework using known separation routines. Nonetheless, the resulting ILP seems to be too large even for relatively small graphs, and it is therefore mainly of theoretical interest.

7 Experiments

We implemented our algorithms as part of the open-source *Open Graph Drawing Framework* (OGDF, [13]). We conducted two series of experiments on a Windows PC with a Pentium 4 (3.4 GHz) processor and 2 GB RAM.

The first experiment uses the well-known *Rome* benchmark set [1], which has been used for many studies on the traditional crossing number, e.g., [4, 7]. It consists of 11528 real-world graphs with 10–100 nodes. We restricted ourselves to the 8013 non-planar graphs with at least 30 nodes, which have an average density of 1.34. Our main focus was to investigate how the minor crossing number compares to the traditional crossing number in real-world settings. Fig. 4 shows the average crossing numbers and minor crossing numbers per graph size. We can see that the minor crossing minimization leads to roughly 35% less crossings on average. While this diagram shows the results for variable embeddings, the diagram looks nearly identical when considering random fixed embeddings, although the absolute crossing numbers are of course a bit higher. In both cases, for

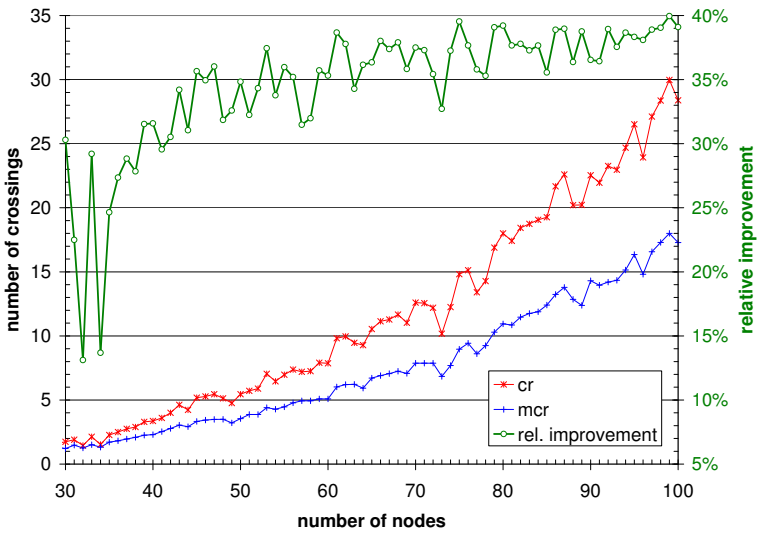


Fig. 4. Results for the Rome graphs (variable embedding).

the large graphs, the realizing graphs have about 10% more nodes then the original graphs, and roughly 8% of the graphs' nodes are substituted by expansion trees. All graphs can be solved clearly under a second for the fixed embedding case and under 30 seconds for the variable case. For the latter, the 100-node graphs required 5.5 seconds on average.

The second set of experiments deals with hypergraphs. Therefore we chose all hypergraphs from the ISCA'85, '89, and '99 benchmark sets of real-world electrical networks with up to 500 nodes in their point-based expansion. The following table summarizes our heuristic results for these graphs, considering both phcr, using our traditional crossing minimizer [7], and thcr. The times are given in seconds. We can clearly see the benefit of considering the tree-based drawing style, as compared to the relatively large point-based crossing numbers.

Other papers like [5] considered the tree-based drawing style with certain additional constraints, in particular requiring certain nodes to lie on the "outside" of the drawing. Hence our solutions are not directly comparable. Nonetheless, we think that the numbers show how promising our approach is: the common graphs *s298* and *s400* required 428 and 400 crossings, respectively, using the best algorithm in [5], while our best solutions are 69 and 86, respectively. Hence it seems worthwhile to investigate how to include such constraints into our algorithm.

References

1. 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.

				FIX				VAR			
name	$\|V\|$	$\|\mathcal{F}\|$	$\|E(H)\|$	phcr	time	thcr	time	phcr	time	thcr	time
b01	43	47	128	32	0.17	18	0.16	33	6.36	18	4.94
b02	25	27	73	12	0.05	8	0.06	12	0.78	8	0.59
b03	148	156	432	148	1.86	58	1.33	131	3:26.91	54	1:32.09
b06	42	50	134	54	0.33	26	0.28	55	13.44	24	7.80
b08	166	179	493	298	8.31	153	4.84	297	18:10.75	146	7:48.16
b09	167	169	472	179	2.27	78	1.77	165	4:47.84	68	2:19.27
b10	183	200	553	431	14.31	212	7.06	432	27:2.30	199	13:42.74
c17	4	11	16	0	0.02	0	0.00	0	0.00	0	0.00
c432	153	196	489	312	6.91	178	4.47	297	15:09.39	167	7:27.97
c499	170	243	578	452	21.19	206	9.78	454	42:55.81	197	20:05.69
s208a	111	122	300	28	0.42	19	0.19	26	15.81	16	9.19
s27a	12	17	33	0	0.00	0	0.00	0	0.00	0	0.00
s298	127	136	385	205	2.97	76	2.14	193	5:42.23	69	2:27.92
s344	164	184	448	57	1.12	38	0.81	57	2:01.20	35	1:01.39
s349	165	185	453	60	0.89	39	0.69	57	1:56.17	37	46.95
s382	173	182	500	206	4.14	88	2.49	186	11:17.64	81	3:58.50
s386a	158	172	511	897	3:22.22	258	16.25	850	147:30.55	238	23:40.38
s400	177	186	518	236	4.53	90	2.55	220	10:06.03	86	3:17.03
s420a	233	252	632	82	2.08	54	1.33	75	4:06.20	47	2:16.05
s444	196	205	569	213	3.77	76	2.11	213	8:31.56	77	3:06.83
s510	210	236	640	1159	2:26.14	489	1:01.41	1096	182:57.00	447	67:21.91
s526a	209	218	675	614	51.83	239	10.39	588	104:08.84	225	27:39.41

2. D. Bokal, É. Czabarkab, L. A. Székely, and I. Vrt'o. Graph minors and the crossing number of graphs. In *Proc. of 6th Czech-Slovak International Symposium on Comb., Gr. Th., Alg. and App.*, volume 28 of *ENDM*, pages 169–175, 2007.
3. D. Bokal, G. Fijavz, and B. Mohar. The minor crossing number. *SIAM Journal on Discrete Mathematics*, 20:344–356, 2006.
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*, 2007. to appear. A preliminary version appeared in *Proc. GD '05*, LNCS 3843, pp. 37–48.
5. T. Eschbach, W. Günther, and B. Becker. Orthogonal hypergraph drawing for improved visibility. *JGAA*, 10(2):141–157, 2006.
6. M. R. Garey and D. S. Johnson. The rectilinear steiner tree problem is np-complete. *SIAM J. Appl. Math.*, 32:826–834, 1977.
7. C. Gutwenger and P. Mutzel. An experimental study of crossing minimization heuristics. In *Proc. GD '03*, volume 2912 of *LNCS*, pages 13–24, 2004.
8. C. Gutwenger, P. Mutzel, and R. Weiskircher. Inserting an edge into a planar graph. *Algorithmica*, 41(4):289–308, 2005.
9. P. Hliněný. Crossing number is hard for cubic graphs. *J. of Combinatorial Theory ser. B*, 96:455–471, 2006.
10. D. S. Johnson and H. O. Pollak. Hypergraph planarity and the complexity of drawing venn diagrams. *J. Graph Theory*, 11(3):309–325, 1987.
11. H. Klemetti, I. Lapinleimu, E. Mäkinen, and M. Sieranta. A programming project: Trimming the spring algorithm for drawing hypergraphs. *SIGCSE Bulletin*, 27(3):34–38, 1995.
12. E. Mäkinen. How to draw a hypergraph. *I.J. Computer Math.*, 34:177–185, 1990.
13. *OGDF – Open Graph Drawing Framework*. Website under construction, 2007.
14. G. Sander. Layout of directed hypergraphs with orthogonal hyperedges. In *Proc. GD '03*, volume 2912 of *LNCS*, pages 381–386, 2004.
15. I. Vrt'o. Crossing numbers of graphs: A bibliography. See `ftp://ftp.ifi.savba.sk/pub/imrich/crobib.pdf`, 2007.