The Constrained Crossing Minimization Problem

Petra Mutzel and Thomas Ziegler*

Max-Planck-Institut für Informatik, Saarbrücken,
{mutzel,tziegler}@mpi-sb.mpg.de

Abstract. In this paper we consider the *constrained crossing minimization problem* defined as follows. Given a connected planar graph $G = (V, E)$, a combinatorial embedding $\Pi(G)$ of G , and a set of pairwise distinct edges $F \subseteq V \times V$, find a drawing of $G' = (V, E \cup F)$ such that the combinatorial embedding $\Pi(G)$ of G is preserved and the number of edge crossings is minimized. The constrained crossing minimization problem arises in the graph drawing method based on planarization.

In [4] we have shown that we can formulate the constrained crossing minimization problem as an $|F|$ -pairs shortest walks problem, where we want to minimize the sum of the lengths of the walks plus the number of crossings between the walks.

Here we present an integer linear programming formulation (ILP) for the *shortest crossing walks problem*. Furthermore, we will present additional valid inequalities that strengthen the formulation. Based on our results we have designed and implemented a branch and cut algorithm. Our computational experiments for the constrained crossing minimization problem on a benchmark set of graphs ([1]) are encouraging. This is the first time that practical instances of the constrained crossing minimization problem can be solved to provable optimality.

1 Introduction

The constrained crossing minimization problem arises in the area of graph drawing. In graph drawing a small number of edge crossings is one of the most important aesthetic criteria. Unfortunately, the problem of minimizing the number of crossings in a drawing is NP-hard ([2]) and so far no practically efficient algorithm exists, even for small nontrivial graphs.

A good method to draw nonplanar graphs with a small number of crossings is the planarization method. This method proceeds as follows:

- (1) Compute a maximum (or maximal) planar subgraph P of the given graph.
- (2) Determine a combinatorial embedding $\Pi(P)$ of P . A *combinatorial embedding* $\Pi(G)$ of a planar graph $G = (V, E)$ is an equivalence class of the planar drawings of G . It fixes for each vertex $v \in V$ the order of the incident edges in a planar drawing of G .

* Research of second author supported by ZFE, Siemens AG, Munich, Germany.

- (3) Reinsert the edge set F removed in step (1) into the fixed combinatorial embedding $\Pi(P)$ of P such that the number of crossings is minimized.
- (4) Replace the crossings by new artificial vertices, draw the resulting planar graph using a planar graph drawing algorithm, and finally, replace the artificial vertices by crossings again.

Here, we investigate the problem arising in step (3), the reinsertion of the edges into the planar subgraph P preserving the combinatorial embedding $\Pi(P)$. This problem can be formulated as the constrained crossing minimization problem defined above.

We can show that the constrained crossing minimization problem is NP-hard. Our proof is based on the NP-completeness of FIXED LINEAR CROSSING NUMBER, shown in [3]. In practice it is attacked by iterative heuristics using the following observation: If only one edge needs to be inserted, the problem can be solved optimally in polynomial time by computing a shortest path in the combinatorial dual graph extended by some vertices and edges. The heuristics iteratively insert the edges using this dual graph approach. However, the result is not always acceptable and we think that the exact solution of the constrained crossing minimization problem will lead to much nicer drawings.

In [4] we have presented the first step towards an algorithm for solving practical instances of the constrained crossing minimization problem to provable optimality. We have shown that we can formulate the constrained crossing minimization problem as a shortest crossing walks problem, which is of rather combinatorial than geometric nature.

A *walk* in a graph G is an alternating sequence of vertices and edges of G , beginning and ending with a vertex, in which each edge is incident to the two vertices immediately preceding and succeeding it. We denote a walk W between two vertices v_0 and v_l by $W = v_0 e_1 v_1 \dots e_l v_l$. If all the edges of a walk are distinct, we call the walk a *trail*. If all the vertices are distinct, we call the walk a *path*.

Since it is much easier to express a set of paths than a set of walks in terms of linear inequalities, we have investigated the shortest crossing paths problem, which is essentially the shortest crossing walks problem in which the set of walks is replaced by a set of paths (see [4]). However, an optimal solution to the shortest crossing paths problem is, in general, not an optimal solution of the constrained crossing minimization problem (see Section 2).

Here we investigate the shortest crossing walks problem. The basic idea for representing a set of walks in terms of linear inequalities is to use variables for pairs of adjacent edges instead of variables for the edges only. Besides the linear inequalities needed in the ILP, we present additional classes of valid inequalities for the problem. We can solve the separation problem for most of the described inequalities. Hence, we can make use of them within a branch and cut algorithm. Our computational experiments show that our new approach is not only of theoretical but also of practical interest.

This paper is organized as follows. In Section 2 we briefly review the relation between the constrained crossing minimization problem and the shortest

Fig. 1. On the left you see a simple crossing at vertex $v \in V$, on the right you see a distributed crossing with common subsequence $v_0e_1v_1e_2v_2e_3v_3$.

crossing walks problem. In Section 3 we present an ILP for the description of all trails between two distinct vertices of a given graph. We use this ILP in Section 4 to describe an ILP for the shortest crossing trails problem. In this section we also present additional valid inequalities for the shortest crossing trails problem. In Section 5 we apply the ILP of Section 4 to the shortest crossing walks problem. In Section 6 we briefly describe a branch and cut algorithm for the constrained crossing minimization problem and present computational results on a benchmark set of graphs [1].

2 Constrained Crossing Minimization and Shortest Crossing Walks

In this section we briefly review the relation between the constrained crossing minimization problem and the shortest crossing walks problem. More details and a proof of the equivalence can be found in [4]. For the rest of the paper we will then concentrate on the shortest crossing walks problem.

In this paper we consider undirected graphs $G = (V, E)$, which may contain loops and multiple edges. We denote an edge of the graph by $e = uv \in E$, where the order of the two end vertices is arbitrary. By $\delta(v) \subseteq E$ we denote the set of edges incident to vertex $v \in V$. We will use the notation $a - b - c$ in $\delta(v)$ to indicate that the edges a, b, c are pairwise distinct, incident to v , and appear in this order in the embedding $\Pi(G)$ around v .

We define the *shortest crossing walks problem* as follows. Given a weighted planar connected graph $G = (V, E)$ with embedding $\Pi(G)$ and a set $F \subseteq V \times V$ of distinct pairs of vertices of G , called *commodities*, find a set of walks in G with the following properties. There is one walk between s_k and t_k for each commodity $k = (s_k, t_k) \in F$, no walk uses an end vertex of a commodity as an internal vertex, and the sum of the weighted lengths of the walks plus the number of crossings between walks is minimum.

Figure 1 illustrates the two kinds of crossings between walks that can appear, called *simple* and *distributed crossings*. In a simple crossing two walks cross at one vertex and in a distributed crossing they have a common subsequence.

The *combinatorial dual graph* $G^* = (V^*, E^*)$ of a planar graph $G = (V, E)$ with embedding $\Pi(G)$ has a vertex $v^* \in V^*$ for each face f of $\Pi(G)$. Corresponding to each edge $e \in E$ there is an edge $e^* \in E^*$ connecting the two vertices v^*

and w^* corresponding to the faces incident to e . The order of the edges around a vertex v^* in $\Pi(G^*)$ is defined by the order of the edges in G on the boundary of the face corresponding to v^* . This defines the embedding $\Pi(G^*)$ of G^* uniquely.

Given an instance, $(G, \Pi(G), F)$, of the constrained crossing minimization problem we compute the corresponding instance, $(G^*, \Pi(G^*), F^*)$, of the shortest crossing walks problem as follows.

Let G^* be the combinatorial dual graph of G . We call the vertices (edges) of the combinatorial dual graph *regular vertices (edges)*. For every vertex of G that is an end vertex of an edge in F we add a new vertex in the appropriate face of $\Pi(G^*)$. We connect each additional vertex to all vertices on the boundary of the face it was placed in (see Figure 2). The resulting graph G^* is still planar and $\Pi(G^*)$ is uniquely determined by $\Pi(G)$. We call the resulting graph G^* the *extended dual graph*. The set F^* is the set of pairs of additional vertices corresponding to the end vertices of the edges in F . To simplify our description we replace every loop in G^* by a path of length two.

We denote the set of regular vertices (edges) by $V_R^* \subseteq V^*$ ($E_R^* \subseteq E^*$). For each commodity $k = (s_k, t_k) \in F^*$ we denote by $E_k^* \subseteq E^*$ the union of the regular edges and the edges incident to s_k and t_k , $E_k^* = E_R^* \cup \delta(s_k) \cup \delta(t_k)$. E_k^* is the set of edges commodity k is allowed to use. By $\delta^k(v) = E_k^* \cap \delta(v)$ we denote the set of edges in E_k^* incident to vertex $v \in V^*$.

We associate weight 0 with the additional edges, weight $1/2$ with the edges replacing a loop, and weight 1 with all the other edges. Using these weights on the edges the value of a solution of the shortest crossing walks problem is the same as the number of crossings in the corresponding solution of the constrained crossing minimization problem.

Figure 2(a) shows an instance of the constrained crossing minimization problem. Here G is the graph induced by the solid edges and we want to insert the dashed edges with a minimum number of crossings. Figure 2(b) shows G together with its extended dual graph. The vertices of the combinatorial dual graph are drawn as rectangles and the edges as grey solid lines. The additional vertices are 1, 3, 4, 5, 6, 7 and the additional edges are drawn as dashed lines.

This example also shows that considering only paths in the extended dual graph is not sufficient to find an optimal solution of the constrained crossing minimization problem. The optimal solution for this example is the one shown in Figure 2(a). This solution has two crossings. However, there is no path in the extended dual graph corresponding to the edge between 3 and 7. We have to use vertex a twice. The best solution using only paths in the extended dual graph needs three crossings.

3 An ILP for Trails

In this section we present an integer linear program (ILP) describing the set of all trails in a graph $G = (V, E)$ between two distinct vertices $s, t \in V, s \neq t$.

We assume that there is no edge between s and t . Moreover, we do not use s and t as internal vertices of the trail. They only appear as first, respectively last,

Fig. 2. Example of a graph and its extended dual graph.

vertex in the trail. These assumptions are naturally satisfied for our shortest crossing walks problem. We also assume that our graph has no loops.

To describe the trails between s and t we introduce a binary variable y_{ef}^v for every pair of adjacent edges $e, f \in \delta(v), v \in V \setminus \{s, t\}, e \neq f$, with

$$y_{ef}^v = \begin{cases} 1, & \text{if } evf \text{ or } fve \text{ appears in the trail,} \\ 0, & \text{otherwise.} \end{cases}$$

We say that a variable y_{ef}^v covers the edges e and f . We need the index v , if e and f are parallel, i.e., they have both end vertices in common.

Every trail starts in vertex s and does not meet s again. Therefore each trail uses exactly one edge incident to vertex s . For the same reason each trail uses exactly one edge incident to t . We can express these properties by

$$\sum_{e=sv \in E} \sum_{f \in \delta(v) \setminus \delta(s)} y_{ef}^v = 1 \text{ and } \sum_{e=tv \in E} \sum_{f \in \delta(v) \setminus \delta(t)} y_{ef}^v = 1. \quad (1)$$

To ensure that each edge of the graph can be used at most once, we use

$$0 \leq \sum_{f \in \delta(v) \setminus \{e\}} y_{ef}^v \leq 1 \quad \forall e = uv \in E; u, v \in V \setminus \{s, t\}. \quad (2)$$

If an edge $e = uv \in E, u, v \in V \setminus \{s, t\}$, is used by a trail, then there is also an edge preceding e and an edge succeeding e in the trail, that is, e has exactly one neighbor $f \in \delta(u) \setminus \{e\}$ at u in the trail and exactly one neighbor $g \in \delta(v) \setminus \{e\}$ at v . To guarantee this property we have

$$\sum_{f \in \delta(u) \setminus \{e\}} y_{ef}^u = \sum_{f \in \delta(v) \setminus \{e\}} y_{ef}^v \quad \forall e = uv \in E; u, v \in V \setminus \{s, t\}. \quad (3)$$

To complete our ILP we also need integrality constraints on the variables

$$y_{ef}^v \in \{0, 1\} \quad \forall e, f \in \delta(v), v \in V \setminus \{s, t\}, e \neq f. \quad (4)$$

The correctness of the ILP can be seen as follows. A trail $T = se_1v_1 \dots e_lt$ between s and t defines in a natural way the values for the variables. We assign value one to the variables $y_{e_ie_{i+1}}^{v_i}$, for $i = 1, \dots, l-1$, corresponding to subsequences e_ie_{i+1} in T , and value zero to all other variables. It is not difficult to check that these variables satisfy all constraints.

Given a feasible integral solution for the ILP, we can determine the corresponding trail as follows. By (1), we have exactly one variable y_{ef}^v with value one among all variables covering an edge $e = sv \in E$ incident to s and $f = vw \in \delta(v) \setminus \delta(s)$. This defines the first and the second edge of the trail, e and f , respectively. By (2) and (3) there is exactly one variable y_{fg}^w with value one for all $g \in \delta(w) \setminus \{f\}$. This defines the third edge g of the trail. Using the variables with value one in this way, we can always find the “next edge” of the trail. This process can only stop at t . Notice however, that feasible solutions can contain additional cycles not reachable from vertex s or t .

Notice also that any feasible solution of the ILP describes a trail between s and t uniquely. We can, in particular, determine locally for each vertex the pairs of incident edges, which are used successively by the trail. This is an important property for our description of crossings between trails.

4 An ILP for the Shortest Crossing Trails Problem

We use the following ILP to describe an instance, $(G, \Pi(G), F)$, of the shortest crossing trails problem.

To describe the trails for commodity $k = (s_k, t_k) \in F$, we define for every pair of adjacent edges $e, f \in \delta^k(v), v \in V_R, e \neq f$, a binary variable $y_{ef}^{vk} \in \{0, 1\}$ and use the constraints of Section 3 on these variables.

To describe crossings between trails we introduce a binary variable z_v^{kl} for every pair of commodities $k, l \in F, k \neq l$, and every regular vertex $v \in V_R$ with

$$z_v^{kl} = \begin{cases} 1, & \text{if trails for } k \text{ and } l \text{ cross at vertex } v, \\ 0, & \text{otherwise.} \end{cases}$$

Let $k, l \in F, k \neq l$, be two commodities. To model simple crossings we introduce at every regular vertex $v \in V_R$ the following constraints

$$z_v^{kl} \geq y_{ac}^{vk} + y_{bd}^{vl} - 1 \quad \forall a, c \in \delta^k(v); b, d \in \delta^l(v); a - b - c - d \text{ in } \delta(v).$$

Such an inequality models the particular simple crossing between commodity k and l at vertex v , where the trail for commodity k uses a and c successively at v , and the trail for commodity l uses b and d successively at v (cf. Figure 1).

For distributed crossings we have the following constraints for every trail $T = v_0e_1v_1 \dots e_pv_p$ with $v_0, \dots, v_p \in V_R$.

$$\sum_{v_i \in T} z_{v_i}^{kl} \geq y_{ae_1}^{v_0k} + \sum_{i=1}^{p-1} y_{e_ie_{i+1}}^{v_i k} + y_{e_pc}^{v_pk} + y_{de_1}^{v_0l} + \sum_{i=1}^{p-1} y_{e_ie_{i+1}}^{v_i l} + y_{e_pb}^{v_pl} - 2p - 1$$

$$\begin{aligned} & \forall a \in \delta^k(v_0), c \in \delta^k(v_p), b \in \delta^l(v_p), d \in \delta^l(v_0), \\ & e_1 - d - a \text{ in } \delta(v_0), \text{ and } e_p - b - c \text{ in } \delta(v_p). \end{aligned}$$

As for simple crossings such an inequality models one particular distributed crossing between the commodities k and l with common subsequence T .

Some more explanation may be necessary here. Consider the distributed crossing shown in Figure 1. If we have this crossing in our solution, we want the sum of the crossing variables of the vertices v_0, v_1, v_2, v_3 to have value one. Therefore the left hand side sums over these variables. The right hand side of the inequality has at most value one (8 variables, each with value at most one). It has value one, when all variables on the right hand side have value one. In all other cases it is nonpositive and the inequality is trivially satisfied. If the right hand side has value one, we must have the distributed crossing in our solution. In this case also the left hand side must have value one to satisfy the inequality.

Our objective is to minimize the sum of the weighted lengths of the trails plus the number of crossings between trails.

The complete ILP looks as follows:

$$\min \sum_{k \in F} \sum_{v \in V_R} \sum_{e, f \in \delta^k(v), e \neq f} \frac{1}{2}(c_e + c_f)y_{ef}^{vk} + \sum_{v \in V_R} \sum_{k, l \in F, k \neq l} z_v^{kl} \quad (5)$$

s.t.

$$y_{ef}^{vk} \in \{0, 1\} \quad \forall k \in F; e, f \in \delta^k(v), v \in V_R, e \neq f \quad (6)$$

$$z_v^{kl} \in \{0, 1\} \quad \forall v \in V_R; k, l \in F, k \neq l \quad (7)$$

$$\sum_{e=s_k v \in E_k} \sum_{f \in \delta^k(v) \setminus \delta(s_k)} y_{ef}^{vk} = 1 \quad \forall k = (s_k, t_k) \in F \quad (8)$$

$$\sum_{e=t_k v \in E_k} \sum_{f \in \delta^k(v) \setminus \delta(t_k)} y_{ef}^{vk} = 1 \quad \forall k = (s_k, t_k) \in F \quad (9)$$

$$0 \leq \sum_{f \in \delta^k(v) \setminus \{e\}} y_{ef}^{vk} \leq 1 \quad \forall k \in F; e = uv \in E_k; u, v \in V_R \quad (10)$$

$$\sum_{f \in \delta^k(u) \setminus \{e\}} y_{ef}^{uk} = \sum_{f \in \delta^k(v) \setminus \{e\}} y_{ef}^{vk} \quad \forall k \in F; e = uv \in E_k; u, v \in V_R \quad (11)$$

$$\begin{aligned} z_v^{kl} &\geq y_{ac}^{vk} + y_{bd}^{vl} - 1 \\ &\forall k, l \in F, k \neq l; v \in V_R; \\ &a, c \in \delta^k(v); b, d \in \delta^l(v); a - b - c - d \text{ in } \delta(v) \end{aligned} \quad (12)$$

$$\sum_{v_i \in T} z_{v_i}^{kl} \geq y_{ae_1}^{v_0 k} + \sum_{i=1}^{p-1} y_{e_i e_{i+1}}^{v_i k} + y_{e_p c}^{v_p k} + y_{de_1}^{v_0 l} + \sum_{i=1}^{p-1} y_{e_i e_{i+1}}^{v_i l} + y_{e_p b}^{v_p l} - 2p - 1$$

$$\forall T = v_0 e_1 \dots e_p v_p \text{ trail, with } v_i \in V_R, \text{ for } i = 0, \dots, p;$$

$$k, l \in F, k \neq l;$$

$$a \in \delta^k(v_0); c \in \delta^k(v_p); b \in \delta^l(v_p); d \in \delta^l(v_0);$$

$$e_1 - d - a \text{ in } \delta(v_0); e_p - b - c \text{ in } \delta(v_p) \quad (13)$$

The correctness of the ILP can be seen as follows. From the correctness of the ILP for trails we know that the variables y_{ef}^{vk} for commodity $k = (s_k, t_k) \in F$ describe a trail between the vertices s_k and t_k . No trail uses an additional vertex as an internal vertex. We need to show the correctness of the crossing constraints and of the objective function.

Consider the constraints for simple crossings. Every such constraint models one particular crossing at a vertex $v \in V_R$ between the commodities k and l . If this crossing appears in our solution, the right hand side of the constraint has value one. In all other cases the value of the right hand side is nonpositive. On the other hand, if one of the simple crossing constraints is tight, we also must have the corresponding simple crossing in our solution. The same argument can be used to show the correctness of the distributed crossing constraints.

The objective function has two parts. The first part computes the sum of the lengths of the trails and the second part computes the number of crossings between trails. By the correctness of the crossing constraints, two trails cross if and only if the sum of the corresponding crossing variables has value one. Therefore the second sum computes the correct value. To see the correctness of the first part consider one commodity $k = (s_k, t_k) \in F$. The variables y_{ef}^{vk} describe a trail from s_k to t_k . Each edge of the trail, except for the first and the last edge, is covered by exactly two variables with value one. Therefore c_e appears twice for each internal edge of the trail and we have to use the coefficient $1/2$. Edges with weight $c_e = 1/2$ are always used in pairs, since one end vertex has degree two. This shows that we compute the length of each trail correctly and the objective function computes the desired value.

We present some classes of valid inequalities now. We do not need these inequalities in our ILP, but they are very useful in our branch and cut algorithm.

Cut Inequalities. Every trail between s_k and t_k must use at least one edge of every cut $C \subseteq E_k$ separating s_k from t_k . We can use this property to define a class of valid inequalities as follows.

$$\sum_{e=uv \in C, v \in V_R} \sum_{f \in \delta^k(v) \setminus \{e\}} y_{ef}^{vk} \geq 1 \quad \forall C \subseteq E_k \text{ } s_k\text{-}t_k\text{-cut} \quad (14)$$

Additional Constraints for Simple Crossings. Let $v \in V_R$ be a regular vertex and $k, l \in F, k \neq l$, be two commodities. Let $\delta(v) = \{e_1, e_2, \dots, e_r\}$ with order $e_1 - e_2 - \dots - e_r$ in $\delta(v)$. Let $D_1 \cup D_2 \cup D_3 \cup D_4 = \delta(v)$ be a partition of $\delta(v)$ with the following properties:

- $D_1 \cap \delta^k(v) \neq \emptyset, D_3 \cap \delta^k(v) \neq \emptyset, D_2 \cap \delta^l(v) \neq \emptyset, D_4 \cap \delta^l(v) \neq \emptyset$.
- For arbitrary $d_1 \in D_1, d_2 \in D_2, d_3 \in D_3, d_4 \in D_4$: $d_1 - d_2 - d_3 - d_4$ in $\delta(v)$.

Then for all $e_k \in D_1 \cap \delta^k(v), e_l \in D_2 \cap \delta^l(v)$ the following inequalities are valid for the shortest crossing trails problem:

$$z_v^{kl} \geq \sum_{f \in D_3 \cap \delta^k(v)} y_{ef}^{vk} + \sum_{f \in D_4 \cap \delta^l(v)} y_{ef}^{vl} - 1 \quad (15)$$

In every integral solution at most one of the variables $y_{e_{kf}}^{vk}$ and at most one of the variables $y_{e_{lf}}^{vl}$ has value one, because all these variables cover the edge e_k , respectively e_l . If both sums have value one, then there is exactly one variable with value one in each sum and we have a simple crossing between k and l at vertex v by the choice of D_1, D_2, D_3 , and D_4 . Otherwise the right hand side of the inequality is nonpositive and the inequality is trivially satisfied.

Additional Constraints for Distributed Crossings. We can use the same idea to get new constraints for distributed crossings. Due to lack of space we can not describe these constraints here. (See [5] for details.)

5 The Shortest Crossing Walks Problem

We want to apply the ILP of Section 4 to the shortest crossing walks problem. The only difference is that walks can use edges of the graph more than once. The idea is to replace every regular edge by an appropriate number of parallel copies (without changing the embedding). Then each time a walk uses a regular edge in an optimal solution of the shortest crossing walks problem it can use a different copy in the new graph and we only have to consider trails.

It is not difficult to see that an optimal solution of the shortest crossing trails problem in the modified graph corresponds directly to an optimal solution of the constrained crossing minimization problem. We can use the same arguments as in our proof of the equivalence between the shortest crossing walks problem and the constrained crossing minimization problem (see [4]). The only problem is to find a bound on the number of copies needed for every regular edge.

The following lemma provides such a bound.

Lemma 1. *Each edge is used at most $r \leq \lfloor \frac{1}{2}(|F|) \rfloor$ times by a walk in an optimal solution of the shortest crossing walks problem.*

Proof. Assume that a walk W uses an edge r times. Then W is at least $2r - 2$ longer than a corresponding shortest path. In an optimal solution of the shortest crossing walks problem two walks can have at most one crossing and therefore the length of every single walk can be at most $|F| - 2$ longer than a corresponding shortest path. So we have $2r - 2 \leq |F| - 2$, that is, $r \leq \lfloor \frac{1}{2}(|F|) \rfloor$.

6 Computational Experiments and Results

Based on the ILP and the valid inequalities we developed a branch and cut algorithm. We start with the constraints (8),(9),(10), and (11) and use the other constraints as cutting planes. We can separate the constraints for simple crossings and cuts exactly in polynomial time. However, our separation of distributed crossing constraints is only a heuristic. (See [5] for more details).

We implemented the algorithm using LEDA [6] and ABACUS [7] and did computational experiments on a benchmark set of graphs [1]. For every graph we computed a planar subgraph and a combinatorial embedding of the planar

subgraph. The benchmark set contains 11529 graphs. In our experiments we considered only the 5157 graphs where we deleted between 2 and 10 edges. We allowed a maximum running time of one hour of CPU-time per instance on a Sun Enterprise 10000 with 12 GB main memory.

We aimed on computing provable optimal solutions for the shortest crossing paths, the shortest crossing trails, and the shortest crossing walks problem on the extended dual graphs of the benchmark graphs. Figure 3(a) shows the percentage of instances we could solve to provable optimality. Whereas we could solve all the instances with 6 commodities of the paths version to optimality, we could only do this for 95% of the instances of the walks version. In the case of 10 commodities, we could compute a provable optimal solutions for 56% of the instances of the paths and for 53% of the trails version, but only 10% of the walks version. The additional inequalities for crossings described in Section 4 are indeed useful in our branch and cut algorithm. Without them we could solve about 12% less instances to optimality within the given time. The average running times per instance is shown in Figure 3(b).

Figure 3(c) shows the number of crossings obtained by the iterative heuristic (described in Section 1), and the result of the branch and cut algorithm for the trails version. The number of crossings improves by about 7% on average compared to the heuristic. The maximal improvement, however, was 33%. Figure 3(c) also shows the trivial lower bound obtained by simply adding the lengths of the shortest paths for the commodities, and the lower bound obtained by our branch and cut algorithm. We also observed that the number of crossings of the paths version and the walks differ only very slightly from that of the trails version. We found 14 instances, where the shortest crossing trails problem finds a better solution than the shortest crossing paths problem, but no examples, where the shortest crossing walks solution improves upon the trails solution.

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.*, 7:303-326, 1997.
2. M.R. Garey and D.S. Johnson. Crossing Number is NP-Complete. *SIAM J. Alg. Disc. Meth.*, Vol.4, No.3: 312-316, 1983.
3. S. Masuda, K. Nakajima, T. Kashiwabara, and T. Fujisawa. Crossing Minimization in Linear Embeddings of Graphs. In: *IEEE Transactions on Computers*, Vol.39, No.1: 124-127, 1990.
4. P. Mutzel and T. Ziegler. The Constrained Crossing Minimization Problem – A First Approach. In: P.Kall, H.-J. Lüthi (Eds.): *Operations Research Proceedings 1998*, Springer-Verlag, pp. 125-134, 1999.
5. P. Mutzel and T. Ziegler. The Constrained Crossing Minimization Problem. Forthcoming *Technical Report*, MPI Informatik, Saarbrücken, Germany, 1999.
6. K. Mehlhorn and S. Näher. LEDA: A platform for combinatorial and geometric computing. *Comm. Assoc. Comput. Mach.*, 38:96-102, 1995.
7. M. Jünger and S. Thienel. Introduction to ABACUS - A Branch-And-CUt System. *Technical Report No. 97.263*, Institut für Informatik, Universität zu Köln, 1997, to appear in *Operations Research Letters*, 1999.

Fig. 3. Computational Results: (a) Percentage of instances solved to optimality, (b) Average running time, (c) Average number of crossings.