Minimizing the Stabbing Number of Matchings, Trees, and Triangulations

Sándor P. Fekete* Marco E. Lübbecke* Henk Meijer†

Abstract

The (axis-parallel) stabbing number of a given set of line segments is the maximum number of segments that can be intersected by any one (axis-parallel) line. We investigate problems of finding perfect matchings, spanning trees, simple polygons, or triangulations of minimum stabbing number for a given set of points. The complexity of these problems has been a long-standing open problem; in fact, it is one of the original 30 outstanding open problems in computational geometry on the list by Demaine, Mitchell, and O'Rourke.

We show that minimum stabbing problems are NP-complete. We also show that Jain's iterated rounding technique is applicable for matchings and spanning trees of minimum stabbing number by showing that there is a polynomially solvable LP-relaxation that has fractional solutions with at least one heavy edge. This strongly suggests constant-factor approximations. Our approach uses polyhedral methods that are related to another open problem (from a combinatorial optimization list), in combination with geometric properties. We also demonstrate that the resulting techniques are practical for actually solving problems with up to several hundred points optimally or near-optimally.

Classification: F.2.2 Nonnumerical Algorithms and Problems

1 Introduction

Objective Functions. Typical problems in combinatorial optimization, algorithmic graph theory, or computational geometry deal with minimizing the length of a desired structure: Given a set of points, find a set of line segments of small total length, such that a certain structural condition is maintained. Among the most popular such structures are spanning trees, perfect matchings, or (in a planar geometric setting) triangulations. However, some geometric scenarios motivate other objective functions; one such alternative for measuring the quality of a structure is the total turn cost between adjacent line segments; e.g., see [3].

When dealing with structural or algorithmic properties, one can be more interested in yet another objective function called the *stabbing number*: For a given set of line segments, this is the maximum number of segments that are encountered by any infinite line; if we consider only axis-parallel lines, we get the *axis-parallel stabbing number*. By considering the minimum number of line segments intersected along a ray to infinity, we get the (axis-parallel) *stabbing depth*, which is roughly half the stabbing number. When focusing on the number of objects defined by the line segments, we may consider the closely related *crossing number* and *crossing depth*, which arise from considering the number of connected components of the set of line segments that we have to cross along a line or a ray to reach infinity. In the absence of connected components of collinear segments (which is the case for matchings), the crossing number is equal to the stabbing number.

*Department of Mathematical Optimization, Braunschweig University of Technology, Pockelsstraße 14, D-38106 Braunschweig, Germany. Email: {s.fekete, m.luebbecke}@tu-bs.de.

†Department of Computing and Information Science, Queen's University, Kingston, Ontario, K7L 3N6, Canada. Email: henk@cs.queensu.ca. Partially supported by NSERC while visiting Braunschweig.

Related Work. Stabbing problems have been considered for a number of years. The complexity of many algorithms in computational geometry is directly dependent on the complexity of ray-shooting, which depends directly on the stabbing depth. Agarwal [1] describes several applications of spanning trees with low stabbing depth, among them *ray-shooting* and *implicit point locations queries* (which by themselves have applications in polygon containment, implicit hidden surface removal, polygon placement, etc.). The theoretically best performing data structure for ray tracing in two dimensions is based on a triangulation of the scene; see Hershberger and Suri [11]. Agarwal, Aronov, and Suri [2] investigate the stabbing depth of triangulations in three dimensions, where the stabbed objects are simplices. Held, Klosowski, and Mitchell [10] investigate collision detection in a virtual reality environment; again, we have a dependency on the stabbing number.

Extremal properties of crossing numbers were considered by Welzl [24], who showed that any planar set of n points has a spanning tree with a crossing number of $O(\sqrt{n})$, and there are examples requiring a crossing number of $\Omega(\sqrt{n})$. Another variant is studied by de Berg and van Kreveld [5]: The stabbing number of a decomposition of a rectilinear polygon P into rectangles is the maximum number of rectangles intersected by any axis-parallel segment that lies completely inside of P; they prove that any simple rectilinear polygon with n vertices admits a decomposition with stabbing number $O(\log n)$, and they give an example of a simple rectilinear polygon for which any decomposition has stabbing number $\Omega(\log n)$. They generalize their results to rectilinear polygons with rectilinear holes. Shewchuk [22] shows that in d dimensions, a line can stab the interiors of $\Theta(n^{\lceil d/2 \rceil})$ Delaunay d-simplices. This implies, in particular, that a Delaunay triangulation in the plane may have linear stabbing number.

Despite of this interest by a fair number of noteworthy researchers, there have been no results or conjectures whatsoever on the *complexity* of stabbing problems. In fact, resolving the complexity of stabbing problems has been one of the original 30 outstanding open problems of computational geometry on the list that was first presented by Mitchell and O'Rourke in [18], based on many years of preceding discussions and informal surveys. An up-to-date and expanded list is maintained by Demaine, Mitchell, and O'Rourke and can be found on the internet [6]. In addition to the open complexity status, the reader should take note that stabbing problems have defied all attempts for obtaining good combinatorial lower bounds, and nothing is known about approximation, other than the factor $O(\sqrt{n})$ that can be deduced from Welzl's work.

Our Work. In this paper, we present the first general algorithmic study of stabbing problems. We resolve the open problem of complexity for various structures; our technique is quite general, and it seems clear that it can be extended to other versions. We also describe a general technique based on Linear Programming (LP) that yields good lower bounds, and is closely related to an open problem from another prominent list, this time from the combinatorial optimization community [14, 15]. As a consequence, we are able to deduce an extension of the Jain's iterated rounding technique for generalized Steiner network problems [12]; making additional use of the geometric structure of the problem, we get the basis for constant-factor approximations. Finally, we demonstrate by a computational study on various benchmark sets that our LP-based approach is also practically useful, both for solving problems approximately or optimally, by considering integer programming (IP).

Our results in detail:

- We prove that deciding whether a point set has a perfect matching of axis-parallel stabbing number 5 is an NP-complete problem; we also extend this result to general stabbing number.
- We sketch NP-completeness proofs for finding a spanning tree of axis-parallel or general stabbing number, and hint at a hardness proof for axis-parallel or general stabbing number.
- We prove that finding a triangulation of minimum axis-parallel crossing number is NP-hard.
- We describe an LP-based class of lower bounds that can be evaluated in polynomial time. From a theoretical point of view, we use the ellipsoid method; the existence of a strongly polynomial algorithm for a closely related class of problems is subject to an open problem from the list [14, 15].

- We give results on the structure of fractional vertices of the resulting LP-relaxation: For matching, we show that there always is an edge with weight at least 1/5, while for spanning trees, there always is an edge with weight greater than 1/3. This allows applying Jain's iterated rounding technique for generalized Steiner network problems and should imply constant-factor approximations.
- We describe the results of a computational study. Using a diverse set of benchmark instances (based on TSPLIB, Solomon's vehicle routing problems, and two different types of random instances) we are able to compute optimal and near-optimal solutions for instances up to several hundred points. This demonstrates that our LP-based approach is good not only in theory (where we get a polynomial running time based on the ellipsoid method), but also in practice (where we use the simplex method). Results indicate far better approximation quality than the theorically possible factors of 5 or 3, respectively.

The rest of this paper is organized as follows. After some basic definitions and notation in Section 2, we give details or sketches of our various hardness proofs in Section 3, with details in Appendix A. In Section 4, we describe our LP-based approach for constructing bounds. Section 5 presents an iterated rounding technique for matching and spanning tree problems; the resulting algorithms appear to be constant-factor approximations. Section 6 presents a detailed computational study on perfect matchings of low stabbing number. Final concluding thoughts and miscellaneous results and problems are presented in Section 7.

2 Preliminaries

In the following, we consider a planar point set P of cardinality n. When dealing with matchings, we assume that n is even, if necessary by omitting one of the points.

Given a set of line segments L, the *stabbing number* of a line ℓ is the number of segments of L that are intersected by ℓ. The *stabbing number of* L is the maximum stabbing number over all lines ℓ; the *axis-parallel stabbing number of* L is the maximum stabbing number over all axis-parallel lines ℓ. In the rest of this paper, the set L will arise as a matching, spanning tree, or triangulation of a planar point set P, and our objective is to find such a structure of minimum stabbing number. We denote by $\textit{St-Mat}_{all}(P)$ the minimum stabbing number among all matchings of P, and by $\textit{St-Mat}_2(P)$ the minimum axis-parallel stabbing number of all matching of P. Similarly, we denote by $\textit{St-Tre}_{all}(P)$ the minimum stabbing number of all spanning trees of P, by $\textit{St-Tre}_2(P)$ the minimum axis-parallel stabbing number of all spanning trees of P; by $\textit{St-}\Delta_{all}(P)$ the minimum stabbing number of all triangulations of P, and by $\textit{St-}\Delta_2(P)$ the minimum axis-parallel stabbing number of all triangulations of P.

Closely related to the stabbing number is the *stabbing depth* of a point p relative to a set L of line segments: This is the minimum number of segments that have to be crossed by a ray from p to reach infinity. The stabbing depth of L is the maximum stabbing depth over all $p \in \mathbb{R}^2$.

If we are not interested in the number of line segments in L encountered by a line ℓ, but in the number of connected components of L encountered by ℓ, we get the *crossing number*. When considering matchings, trees, and triangulations, we may use the analogous abbreviations $\textit{Cr-Mat}_{all}(P)$, $\textit{Cr-Mat}_2(P)$, $\textit{Cr-Tre}_{all}(P)$, $\textit{Cr-Tre}_2(P)$, $\textit{Cr-}\Delta_{all}(P)$, $\textit{Cr-}\Delta_2(P)$. Furthermore, *crossing depth* arises by considering the number of connected components of L encountered by a ray. Note that stabbing and crossing number or depth are the same for matchings.

Interestingly, stabbing depth of matchings is a combinatorial, not a geometric property.

3 Complexity

In this section we give proofs and proof sketches that virtually all variants of minimum stabbing problems are NP-hard. Our technique is rather general and should be applicable to other variants as well. In this extended abstract, we give only proof sketches; full details of the proofs for the stabbing number of matchings and

the crossing number of triangulations can be found in the appendix; for trees, proofs are analogous, so we omit full details.

3.1 Matchings

Theorem 1 *Deciding whether St-Mat$_2(P) \leq 5$ is strongly NP-complete.*

Proof. See Appendix A for full details. We prove the theorem using a reduction from 3SAT [9]. Assume we have a Boolean expression $B(x_0, x_1, \ldots, x_{n-1})$ with n variables and k clauses of three literals. We construct a set of points P that has a matching M of stabbing number 5 if and only if the Boolean expression can be satisfied. For the overall layout see Figure 1. Note the collinear sets of points that function as "barriers": As they already require a stabbing number of 5, they must not be crossed by additional line segments, thus imposing a clear combinatorial structure that is exploited in the proof.

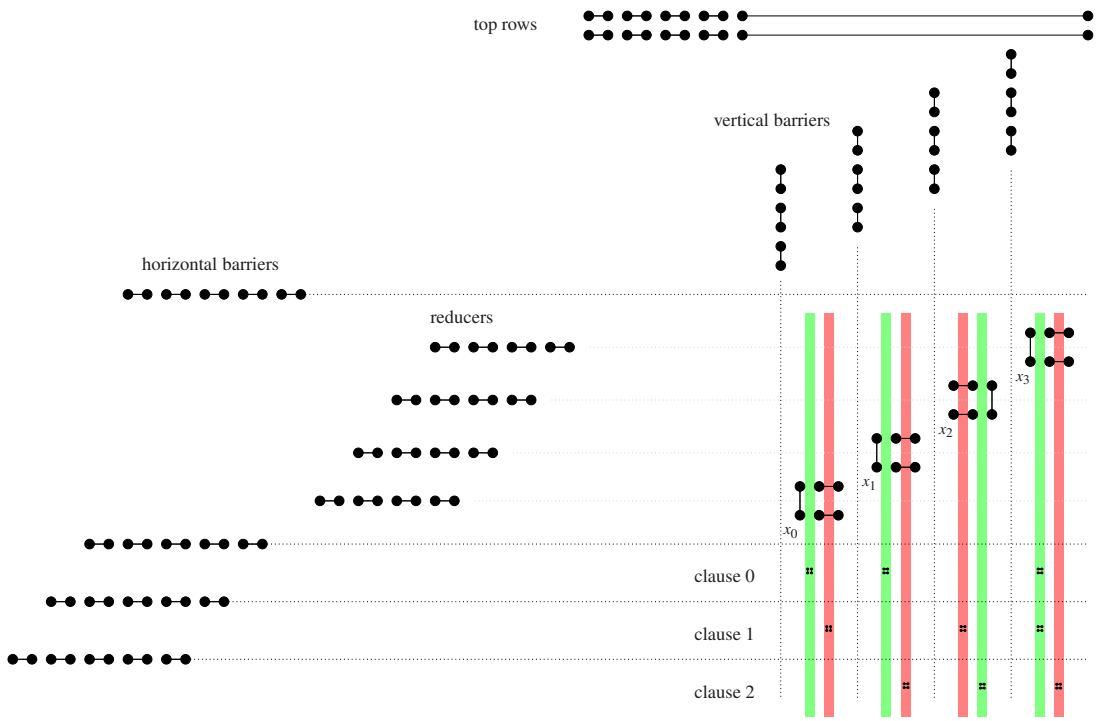

Figure 1: Overall layout for *St-Mat*$_2(P)$.

Figure 8 shows a set of 6 points, numbered i_0 to i_5 in clockwise order, that represents a variable.

Each variable gadget allows two feasible matchings; the particular choice represents a truth assignment for a particular variable, which in turn imposes requirements on how the literal gadgets in the respective columns may be chosen. As shown in Figure 9, clauses are represented by three literal gadgets in the same row; the overall construction implies that the stabbing number of the row of a clause is at most five, iff at least one literal gadget in this row contributes only one stabbed line segment, meaning that at least one literal satisfies the clause. □

Corollary 2 *There is no* α*-approximation algorithm for St-Mat*$_2(P)$ *with* $\alpha < 6/5$.

Corollary 3 *Computing St-Mat*$_{all}(P)$ *is weakly NP-hard.*

Proof. We apply a perturbation technique, similar to the one in [8]. Use the same construction as for the hardness proof for the axis-parallel case. Consider the grid formed by the coordinates of the resulting point set. This grid is modified such that the interpoint distances between the points of the same gadget are $\Theta(\varepsilon^{n^2})$. Furthermore, the rest of the grid is perturbed by powers of ε, such that only axis-parallel lines can stab more than two gadgets. Now it is easy to see that only axis-parallel lines are critical. □

3.2 Spanning Trees

The basic construction for showing hardness of finding a spanning tree of minimum stabbing number is very similar to the one for matchings. Therefore, we omit detailed proofs from this short abstract. As before, we use barriers to restrict possible connections: We make use of the arrangement shown (in horizontal orientation) in Figure 2.

Lemma 4 *Let S be the arrangement of $3k$ points shown in Figure 2, and let $P \supseteq S$ have no other points in the horizontal strip indicated by shading. If P has a spanning tree T with stabbing number $k+1$, then no edge of T crosses the shaded region.*

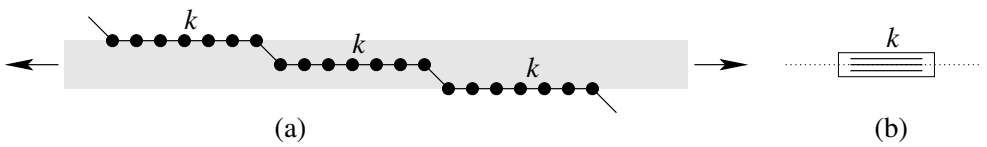

Figure 2: A barrier gadget: (a) In a spanning tree of stabbing number $k+1$, no line segment may cross the shaded region. (b) Symbol for the barrier gadget; the dotted line indicates the blocked strip.

The variable gadgets look as in Figure 3. Note the use of barrier gadgets, and the remaining numbers of segments that may cross the induced dotted lines.

Lemma 5 *Let S be the arrangement of points shown in Figure 3, with barrier gadgets placed and sized as indicated, and let $P \supseteq S$. Then any spanning tree of P that has stabbing number at most $k+1$ uses either the two "true" or the two "false" edges.*

The overall layout of the construction is shown in Figure 4. For proving hardness of finding a spanning tree of minimum crossing number we use the following barrier gadget. For a proof note that each elementary 2×2 cycle of the gadget must have at least one edge not present in a tree; then the claim follows by pigeonhole principle. Finally, the NP-hardness proof for all directions follows again by making only the axis-parallel directions to be critical.

Lemma 6 *Let S be the $k \times (k-1)^2 + k$ arrangement of points shown in Figure 5, and let $P \supseteq S$. If P has a spanning tree T with crossing number k, then no other edge of T crosses the shaded region.*

Summarizing, we state:

Theorem 7 *Finding St-Tre$_2(P)$, St-Tre$_{all}(P)$, Cr-Tre $_2(P)$, Cr-Tre $_{all}(P)$ are all NP-hard probems.*

3.3 Triangulations

We use the following terminology. A *horizontal line* is a set of points that are on a horizontal line. A *vertical line* is a set of points that are on a vertical line. A *row* consists of two horizontal lines, and the (empty) space between them. A *column* consists of two vertical lines, and the (empty) space between them.

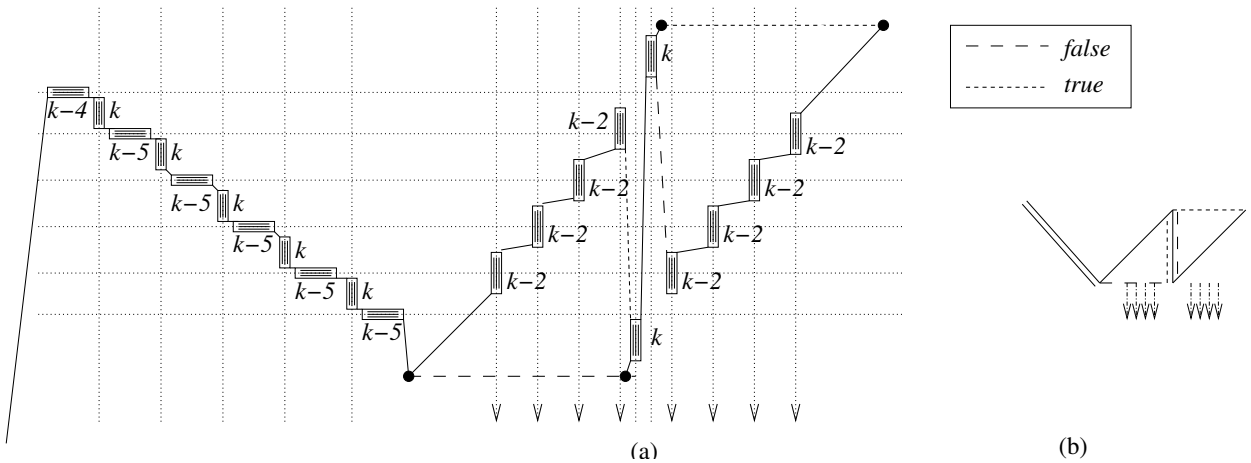

Figure 3: A variable gadget: (a) In a spanning tree of stabbing number $k+1$, either the "true" or the "false" setting is chosen. (b) Symbol for the variable gadget.

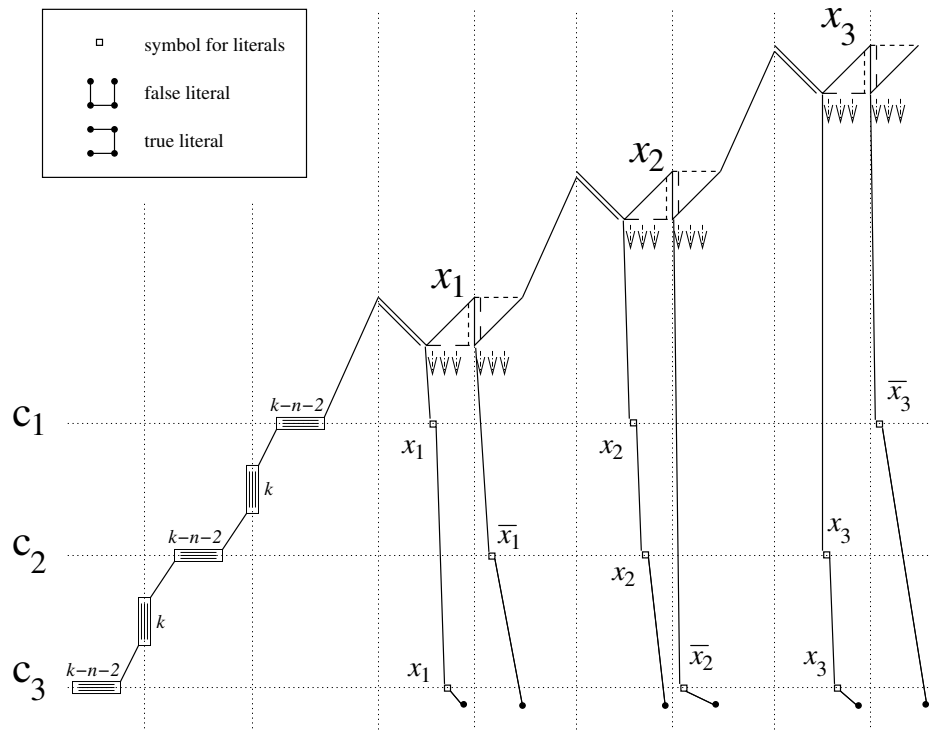

Figure 4: The overall layout for the hardness proof for spanning trees. n is the total number of literals, k a sufficiently large number, the existence of a spanning tree with stabbing number $k+1$. Shown is the representation of the 3SAT instance $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3)$.

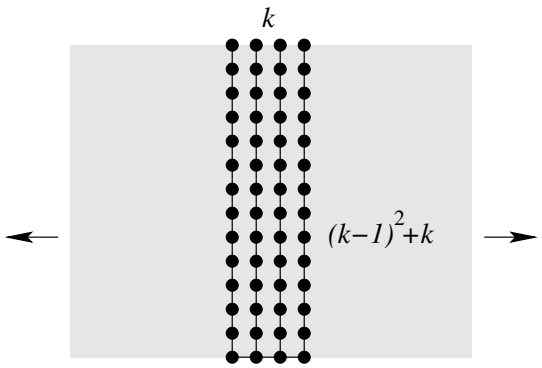

Figure 5: A barrier gadget for showing hardness of minimizing the crossing number of a spanning tree.

Lemma 8 *Consider a row consisting of two horizontal lines in P, having a and b points, respectively. The horizontal stabber of this row encounters at least $a+b-2$ triangles in any triangulation of P.*

A proof can be found in Appendix A. The lemma analogously holds for two vertical lines that form a column. When a row consists of two horizontal lines that have $Cr\text{-}\Delta_2(P)+2$ points altogether, we call it *full* or fully triangulated. In a triangulation that achieves the lower bound of the lemma this row has the property that all edges along the horizontal lines have to be present. The same applies to full columns, and this is the way how we will use the lemma.

Theorem 9 *Finding $Cr\text{-}\Delta_2(P)$ is NP-hard.*

The proof is based on another reduction of 3SAT. Full details can be found in Appendix A. See Figure 6 for a schematic layout of a representing point set P for the 3SAT instance $B(x_0,x_1,x_2) = (x_0 \vee x_1 \vee \bar{x}_2) \wedge (x_0 \vee \bar{x}_1 \vee x_2) \wedge (\bar{x}_0 \vee \bar{x}_1 \vee \bar{x}_2)$. Figure 10 shows the structure of variable gadgets.

4 Linear Programs for Stabbing

4.1 Linear Programs and *St-Mat*(P)

A handsome representation of a perfect matching M is by its edge incidence vector $x \in \{0,1\}^E$, where $x_{ij} = 1$ if $ij \in M$ and $x_{ij} = 0$ otherwise. For $S \subseteq V$ denote by $\delta(S) = \{ij \in E \mid i \in S, j \notin S\}$. The following linear inequalities are necessarily satisfied for a perfect matching given by x.

$$\sum_{ij \in \delta(i)} x_{ij} = 1 \quad \forall i \in V \tag{1}$$

$$\sum_{ij \in \delta(S)} x_{ij} \geq 1 \quad \forall S \subseteq V,\ |S| \text{ odd} \tag{2}$$

$$x \geq 0 \tag{3}$$

In a seminal paper Edmonds [7] showed that this polyhedral description is also sufficient in the sense that the extreme points of the polytope Π defined by (1)–(3) are exactly the incidence vectors of perfect matchings in G. Despite the fact that there are exponentially many so-called *blossom inequalities* (2) one can solve a linear program over Π in strongly polynomial time [4].

Now we wish to minimize the stabbing number depth k and add to Π

$$\sum_{ij : ij \cap \ell(d) \neq \emptyset} x_{ij} \leq k \qquad \text{for each stabbing line } \ell(d) \text{ in direction } d. \tag{4}$$

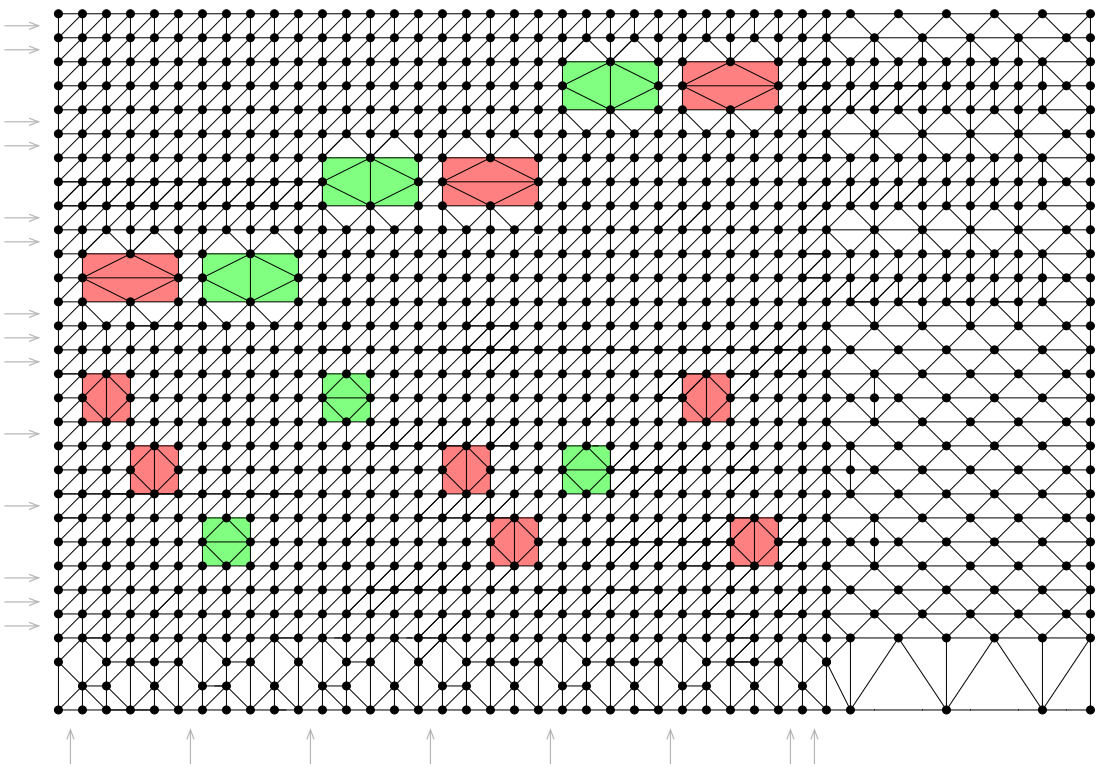

Figure 6: Overall layout for $Cr\text{-}\Delta_2(P)$. Clauses are $(x_0 \vee x_1 \vee \bar{x}_2)$, $(x_0 \vee \bar{x}_1 \vee x_2)$, and $(\bar{x}_0 \vee \bar{x}_1 \vee \bar{x}_2)$, with $x_0 = \mathit{false}$ and $x_1 = x_2 = \mathit{true}$. Arrows indicate full rows and columns.

In principle, there are infinitely many constraints of this type, even for one direction. Note, however, that when sweeping a stabbing line in direction d the stabbing number depth changes only at a vertex. Therefore, we only need to check a linear number of lines in each direction. For the same reason, "all" directions reduce to the $O(n^2)$ combinatorial directions determined by all pairs of vertices of G. When requiring integrality of x and minimizing k in the resulting integer program yields exactly $\mathit{St\text{-}Mat}(P)$. When integrality is relaxed to (3), this linear programming relaxation gives a lower bound on $\mathit{St\text{-}Mat}(P)$. The solution x will in general be fractional, and we speak of *fractional stabbing number* in this context.

The resulting LP can be solved in weakly polynomial time by means of the ellipsoid method [17]: Separating blossoms is possible in polynomial time [19], and there are only polynomially many stabbing constraints. It should be noted that this is closely related to another well-considered open problem: [15, 14] asks for a strongly polynomial algorithm for finding an optimal matching in the presence of a single general "budget" constraint, which is a general version of stabbing constraints. This illustrates that giving a strongly polynomial algorithm for our class of LPs may not an easy task, even though it is the intersection of two well-behaved polyhedra. This difficulty has been known for other classes of intersecting polyhedra [21].

4.2 Linear Programs and $\mathit{St\text{-}Tre}(P)$

There are several polynomial size formulations for spanning trees [16]. We tried a *flow formulation* and a *multicommodity flow formulation*. We will see in the following section that the following cut-based LP formulation has some particularly useful properties:

$$\sum_{ij \in E} x_{ij} = n - 1 \tag{5}$$

$$\sum_{ij \in \delta(S)} x_{ij} \geq 1 \quad \forall S \subseteq V, \tag{6}$$

$$\sum_{ij \in E(S)} x_{ij} \leq |S| - 1 \quad \forall S \subseteq V, \tag{7}$$

$$x \geq 0 \tag{8}$$

with the additional stabbing constraints

$$\sum_{ij : ij \cap \ell(d) \neq \emptyset} x_{ij} \leq k \qquad \text{for each stabbing line } \ell(d) \text{ in direction } d. \tag{9}$$

Again, solving this LP when minimizing k can be achieved in (weakly) polynomial time.

5 Iterated Rounding

Key ingredients of the linear programs described in the previous section are constraints having a *cut structure*: We require that for any set from a given family of subsets of vertices, there is a guaranteed lower bound on the size of a cut. Integer programs of this type have been called *generalized Steiner network problems*. It is known that these problems do not tend to have nicely structured fractional vertices. This does not get better in the presence of stabbing constraints: as our example in Figure 13 illustrates, there may be edges of rather small fractional weight. This prohibits a straightforward approximation by simply rounding up a fractional solution.

A very elegant general technique that overcomes these difficulties and achieves a 2-approximation algorithm for generalized Steiner networks problems was presented by Jain in his paper [12]: Based on a polyhedral argument, he established that any fractional solution of a generalized Steiner network LP must have an edge of weight at least 1/2. From this, he derived an approximation algorithm by iteratively rounding up the weight of the heaviest edges, and re-solving the LP with these fixed edge weights.

It is natural to consider such an approach for our LPs for deriving approximation methods for stabbing problems, in particular for matchings and spanning trees, as the stabbing constraints also have cut structure. However, Jain's crucial lemma guaranteeing a heavy edge in a fractional solution does no longer apply in the presence of stabbing constraints: Figure 7 shows a matching instance for which the optimal fractional axis-parallel stabbing number is achieved by a unique solution with maximum edge weight 1/3.

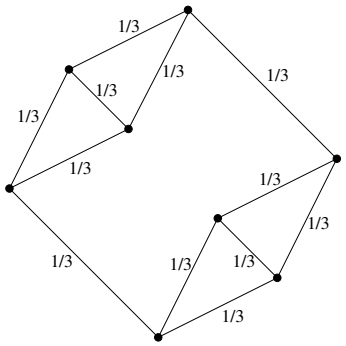

Figure 7: An optimal fractional solution with maximum edge weight 1/3.

However, using additional geometric properties, we can still establish lower bounds. Our proofs are based on the following lemmas.

Lemma 10 *For any even set of points in the plane, there is a fractional perfect matching x of minimum stabbing number, such that the support graph of x is planar. Such a fractional matching can be found in polynomial time.*

Lemma 11 *For any set of points in the plane, there is a fractional spanning tree x of minimum stabbing number, such that the support graph of x is planar. Such a fractional spanning tree can be found in polynomial time.*

Both proof use an uncrossing 2-exchange argument. As the proof for Lemma 10 requires some extra care because of the blossom inequalities, we give a detailed proof in Appendix A. The proof of Lemma 11 is almost completely analogous.

Theorem 12 *For any even set of points in the plane, there is a fractional perfect matching x of minimum stabbing number that has an edge of weight at least 1/5. For any set of points in the plane, there is a fractional spanning tree x of minimum stabbing number that has an edge of weight more than 1/3.*

Proof. For both problems, consider a fractional vertex with a planar support graph. To see the claim for matchings, note that there must be a vertex with maximum degree 5; as the total weight for each vertex is 1, the claim follows. To see the claim for spanning trees, note that the total edge weight is $n-1$, and the number of edges is at most $3n-6$, implying that the average weight is larger than 1/3. □

This sets the stage for an iterated rounding procedure: At each stage, fix the weight of an edge of maximum fractional weight to 1, and resolve the linear program. In each iteration, the number of edges with fractional weight is reduced, so we get an overall polynomial method for finding an integral solution.

It should be noted that even though the key ingredient for an iterated rounding procedure is provided by Theorem 12, there is still one element missing for establishing that these polynomial algorithms are indeed constant-factor approximation algorithms: As the objective function is a maximum, not a sum, we still need an extra argument to assure that Jain's overall approach does indeed work. We are hopeful to complete this argument in the very near future [13]. As we show in the following Section 6, the practical performance seems to be even better than the theoretically possible guarantees of 5 and 3.

6 Computational Results

Test suite. We compiled a test suite of various instances on which we evaluated our LP/IP approaches and the iterated rounding technique for $\textit{St-Mat}_2(P)$. The suite includes ten instances with up to 442 points from the TSPLIB[20]; the C-class ("clustered") of Solomon's instances of the vehicle routing problem [23] with 100 points each; 25 regular grids with 20 to 360 points, based on grids of size 5×5 up to 20×20 in which 20% randomly chosen points are removed; and a set of instances with up to 100 random points in the plane. Tables 1 and 2 in Appendix B display our preliminary results on a Pentium III 700MHz PC with 1GB main memory running Linux.

Some brief observations. In LP solutions variables may assume rather arbitrary fractional and small values; this is also true when blossom inequalities are added. The colinearity of points in the grid instances enables us to reduce the number of stabbing constraints, resulting in significantly reduced computation times. The clustering of points in the vehicle routing instances obviously facilitate the LP/IP solution process, as was to be expected. However, this observation is interesting in practice where the data is usually well structured as opposed to randomly distributed.

The stabbing number obtained from iterated rounding is often very close to the LP lower bound, and in our experiments it is never off the optimal value by more than a factor of 2; typically, it is much better, and mostly within about 20% of the optimum. We observe that the "bad moves" are made only in the final iterations of the iterated rounding. For instance, about 100 iterations are needed for lin318 (with 318 points),

the LP optimum is at 8.113, and the LP value exceeds 9.000 only in the last ten iterations, where a value of 11 is reached. We also experimented with a "one good shot at once" approach that is based on the fact that each fractional vertex is the convex combination of perfect matchings, by finding a maximum weight perfect matching in the support graph of the LP solution. This usually gives even better feasible solutions than for iterated rounding (with one exception in our test suite of problems). This technique certainly deserves further evaluation both from a computational and from a theoretical point of view, and a discussion will be included in the full paper.

The stabbing constraints seem to completely destroy the polyhedral structure of the matching polytope. Half of the TSP instances are infeasible (because of an odd number of points), and this is not detected by the CPLEX IP solver within 4 hours. Iterated rounding terminates in this case with a non-perfect matching with one point unmatched.

Even though our prototypes are not yet ready to solve even larger instances, they may serve as a good starting point for the development of an industrial strength code. More computational results, in particular for $St\text{-}Mat_{all}(P)$ and $St\text{-}Tre_2(P)$, are planned to be included in the full paper. We also plan tests with IP formulations for triangulations.

7 Notes and Conclusion

We have shown that various versions of stabbing problems are NP-hard, and demonstrated how an IP/LP-based approach may be useful for solving and approximating instances. We expect a number of extensions of this work and hope to include more details and results in a forthcoming full journal version. Here we only mention a number of other aspects that are also possible.

Clearly the most interesting open problem is a proof that our iterated rounding technique is indeed a constant-factor approximation algorithm, with a performance guarantee of 5 for matchings, and 3 for spanning trees. There are still some technical difficulties in the related polyhedral discussion, but we are optimistic of completing this proof in the very near future [13].

Another interesting question is to decide the existence of structures of small constant stabbing number. As the hardness proof for deciding the existence of a matching of stabbing number 5 illustrates, this is still not an easy task. From some solvable special cases, we only note one:

Theorem 13 *$St\text{-}Tre_2(P)=2$ and $St\text{-}Mat_{all}(P)=2$ can be decided in polynomial time.*

One may also ask for the *average* instead of the maximum stabbing number, and refer to the average over all possible lines intersecting a set of line segments, instead of just a combinatorial set of representatives. This, however, amounts to solving problems of minimum length, with all implications to hardness and approximation.

Theorem 14 *A structure consisting of line segments has minimum average (axis-parallel, resp.) stabbing number, iff the overall Euclidean (Manhattan, resp.) length of all line segments is minimum.*

Finally, the closely related question of investigating maximum stabbing-independent subsets, i.e., subsets of lines with stabbing number 1, may be interesting in its own right. Decompositions into stabbing-independent subsets may also be a combinatorial approach for getting approximation methods for stabbing problems.

Acknowledgments

We thank Joe Mitchell for pushing us to work on this problem, and repeated discussions that assured further progress. We also thank Kamal Jain for some discussions on iterated rounding.

References

[1] P.K. Agarwal. Ray shooting and other applications of spanning trees with low stabbing number. *SIAM Journal on Computing*, 21(3):540–570, 1992.

[2] P.K. Agarwal, B. Aronov, and S. Suri. Stabbing triangulations by lines in 3D. In *Proceedings of the 11th ACM Symposium on Computational Geometry*, pages 267–276, 1995.

[3] E.A. Arkin, M.E. Bender, E.D. Demaine, S.P.Fekete, J.S.B.Mitchell, and S.Sethia. Optimal covering tours with turn costs. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 138–147, 2001.

[4] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. Wiley, New York, 1998.

[5] M. de Berg and M. van Kreveld. Rectilinear decompositions with low stabbing number. *Inform. Process. Lett.*, 52(4):215–221, 1994.

[6] E.D. Demaine, J.S.B. Mitchell, and J. O'Rourke. The open problems project. `http://cs.smith.edu/~orourke/TOPP/Welcome.html`, 2003.

[7] J. Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *J. Res. Nat. Bur. Standards*, 69B:125–130, 1965.

[8] S. P. Fekete. On simple polygonalizations with optimal area. *Discrete and Computational Geometry*, 23:73–110, 2000.

[9] M.R. Garey and D.S. Johnson. *Computers and Intractability—A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco, 1979.

[10] M. Held, J.T. Klosowski, and J.S.B. Mitchell. Evaluation of collision detection methods for virtual reality fly-throughs. In *Proceedings of the 7th Canadian Conference on Computational Geometry*, pages 205–210, 1995.

[11] J. Hershberger and S. Suri. A pedestrian approach to ray shooting: Shoot a ray, take a walk. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 54–63, 1993.

[12] K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[13] K. Jain. Personal communication, 2003.

[14] A. Jüttner. On budgeted optimization problems. *Mathematics of Operations Research*, to appear.

[15] T. Király. EGRES list of open problems in combinatorial optimization. `http://www.cs.elte.hu/egres/problems/prob_22/index2.html`, 2002.

[16] T.L. Magnanti and L.A. Wolsey. Optimal trees. In M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, chapter 9, pages 503–616. North-Holland, 1995.

[17] M.Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, Berlin, 1988.

[18] J.S.B. Mitchell and J. O'Rourke. Computational geometry column 42. *Int. J. Comput. Geom. Appl.*, 11(5):573–582, 2001.

[19] M. Padberg and M.R. Rao. Odd minimum cut-sets and *b*-matchings. *Mathematics of Operations Research*, 7:67–80, 1982.

[20] G. Reinelt. TSPLIB – A traveling salesman problem library. *ORSA J. Comput.*, 3(4):376–384, 1991.

[21] A. Schrijver. Personal communication, 2002.

[22] J.R. Shewchuk. Stabbing delaunay tetrahedralizations. Umpublished Manuscript, 2002. Dept. Electrical Engineering and Computer Science, UC Berkeley, CA.

[23] M.M. Solomon. VRPTW benchmark problems. `http://w.cba.neu.edu/~msolomon/problems.htm`, 2003.

[24] E. Welzl. On spanning trees with low crossing numbers. In B. Monien and Th. Ottmann, editors, *Data Structures and Efficient Algorithms*, volume 594 of *Lecture Notes in Computer Science*, Berlin, 1992. Springer-Verlag.

Appendix A: Proof Details

Proof of Theorem 1

For a Boolean expression $B(x_0, x_1, \ldots, x_{n-1})$ with n variables and k clauses of three literals, we construct a set of points P that has a matching M of stabbing number 5 if and only if the Boolean expression can be satisfied. Consider the layout as shown in Figure 1. At the top of the layout are 2 groups of 10 points. The points in a group of 10 have the same y-coordinates. We call these two groups the *top rows*. The i-th point in the first top row has the same x-coordinate as the i-th point in the second top row. Below the top rows are n groups of 6 points. Each point in a group of 6 has the same x-coordinates, as shown in the figure. The vertical lines through these groups of 6 points separate variables and separate the variable from other points left of the variables. We call each such group a *vertical barrier*. The points in barrier i are far enough below the points in barrier $i+1$ to ensure that horizontal lines through the vertical barriers have stabbing number at most 5. The barriers lie between the last and second last points of the top rows. To the left of the top rows and below the vertical barriers are $k+1$ groups of 10 points. Each point in a group of 10 has the same y-coordinates. The horizontal lines through these groups of 10 points separate clauses. We call each such group a *horizontal barrier*. The points in barrier i are far enough to the right of the points in barrier $i+1$ to ensure that vertical lines through the horizontal barriers have stabbing number at most 5. The horizontal barriers are used to separate clauses from each other, to separate the clauses from the variables, and to separate variables from other points above the variables. Between and to the right of the top two horizontal barriers are groups of 8 points. Each point in a group of 8 has the same y-coordinates. We call each such group a *reducer* of a variable. The points in reducer i are far enough to the right of the points in reducer $i+1$ to ensure that vertical lines through the reducers have stabbing number at most 5.

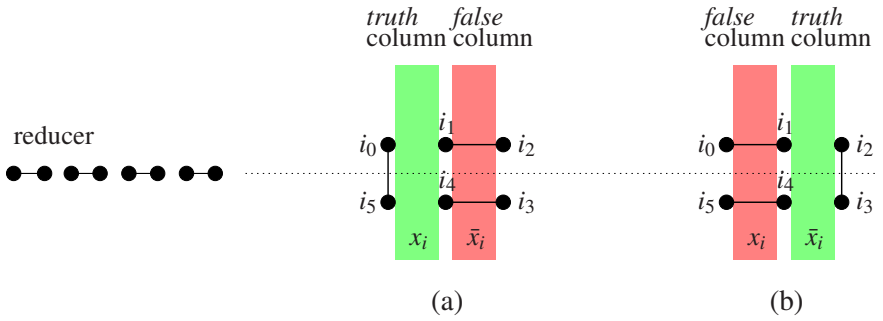

Figure 8: Variable x_i with $x_i = \mathit{true}$ in (a) and $x_i = \mathit{false}$ in (b).

Figure 8 shows a set of 6 points, numbered i_0 to i_5 in clockwise order, that represents a variable. The variables are separated by vertical barriers, and are placed below the vertical barriers, between top two horizontal barriers and to the right of the reducers. Variables are placed such that the y-coordinate of the horizontal line through the reducer of a variable is inbetween the y-coordinates of points i_0 and i_5 of that variable. The collection of vertical lines between points i_0 and i_1 of variable x_i is called the x_i *column* of the variable. The collection of vertical lines between points i_1 and i_2 of variable x_i is called the $\bar{x}_i$ *column* of the variable.

The horizontal and vertical barriers create kn locations for literals. The literals are groups of 4 points representing the presence of a variable in a clause. Each group of 4 points forms an axis oriented square. If a literal x_i appears in the clause c_j, we place it in the x_i column of clause c_j. If a literal $\bar{x}_i$ appears in the clause c_j, we place it in the $\bar{x}_i$ column of clause c_j. The three literals of a clause are put on the same horizontal lines. The literal in column x_i of clause c_j is to the left of the literal in column x_i of clause c_h for $j < h$. Similarly, the literal in column $\bar{x}_i$ of clause c_j is to the left of the literal in column $\bar{x}_i$ of clause c_h for $j < h$. Figure 9 shows the literals of two clauses.

We first assume that $B(x_0, x_1, \ldots, x_{n-1})$ is satisfiable. We will show that P has a matching M of stabbing

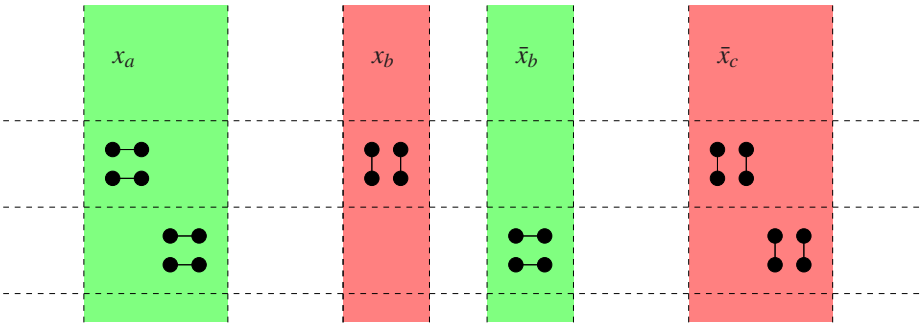

Figure 9: Clauses $(x_a \vee x_b \vee \bar{x}_c)$ and $(x_a \vee \bar{x}_b \vee \bar{x}_c)$ with $x_a = x_c = \textit{true}$ and $x_b = \textit{false}$.

distance 5. We connect point i to point $i+1$ in each of the two top rows for $i = 0,2,4,6,8$. We connect point i to point $i+1$ in each vertical barrier for $i = 0,2,4$. We connect point i to point $i+1$ in each horizontal barrier for $i = 0,2,4,6,8$. We connect point i to point $i+1$ in each reducer for $i = 0,2,4,6$.

If the variable x_i has the value *true* we connect the pairs (i_0,i_5), (i_1,i_2), and (i_3,i_4) of the variable. If the variable x_i has the value *false* we connect the pairs (i_0,i_1), (i_2,i_3), and (i_4,i_5). Notice that if x_i is *true*, then any vertical line in the x_i column stabs 2 edges in the top rows of M, and a vertical line in the $\bar{x}_i$ column stabs 2 edges in the top row and 2 edges in the rectangle of the variable. If x_i is false, this situation is reversed. The column with vertical stabbing number 2 is called the *truth* column of the variable, the column with vertical stabbing number 4 is called the *false* column. In each literal representing the value *true* we connect the four points with two horizontal edges. In each literal representing the value *false* we connect the four points with two vertical edges.

We can now verify that M has stabbing number 5. Any vertical line in the *truth* column of variable x_i stabs two edges in the top rows and at most two edges in a literal. Any vertical line in the *false* column stabs two edges in the top rows, two edges of the variable and at most one in a literal. Any horizontal line in a clause stabs at most three literals, one of which is set to *true*. So these lines stab at most 5 edges of M. It can easily be verified that all other horizontal and vertical lines stab at most 5 edges from M.

Secondly, we assume that P has a matching of stabbing number 5. We show that $B(x_0,x_1,\ldots,x_{n-1})$ is satisfiable. The matching used in this proof is illustrated in Figures 1, 8 and 9. Because the top rows contain 10 points, these points have to be connected to each other, otherwise the stabbing number of P exceeds 5. There are several ways to connect the sets of 10 points. If we connect point i to point $i+1$ for $i = 0,2,4,6,8$ in each row, the number of edges stabbed by any horizontal or vertical line is minimized. Therefore we may assume without loss of generality that these edges are in the matching M. Similarly if in the remainder of this proof there are several ways to connect a set of points, and one of these ways has the minimal stabbing number for all stabbing lines, we will use these connections.

Therefore we can connect point i to point $i+1$ for $i = 0,2,4,6,8$ in each horizontal barrier. For the same reason, we can connect point i to point $i+1$ for $i = 0,2,4,6$ in each reducer. Because vertical lines through the vertical barriers stab two edges in the top rows, we can connect point i to point $i+1$ for $i = 0,2,4$ in each vertical barrier.

Each reducer contributes to a horizontal stabbing number by 4. Thus, we cannot connect the six points of a variable x_i by three vertical edges. Figure 8 shows the two remaining essentially distinct matchings.

Of the three literals in each clause, one has to be set to true, otherwise there will be a horizontal stabber intersecting 6 edges. The true literal, say, x, must lie in a *truth* column of a variable, because vertical lines in this column have stabbing number 2. Any other literal in this column can also be set to *true*. The literals $\bar{x}$ lie in the *false* column of the same variable, and have to be set to false. So if a matching of number 5 exists, there is a truth assignment of the Boolean expression. □

Proof of Lemma 8

Because the row has to be triangulated, there is at least one triangle adjacent to each point. Thus, from each point at least one edge of some triangle must emerge, going to a point at the respective other line. The cheapest way to fulfill this connectedness requirement is to let these edges form a (planar) tree, yielding at least $a+b-1$ edges. Connecting all points consecutively on each respective line then gives $a+b-2$ triangles. □

Proof of Theorem 9

Again, we use a reduction from 3SAT, and the proof proceeds along the lines of the proof of Theorem 1. For a given Boolean expression $B(x_0, x_1, \ldots, x_{n-1})$ with n variables and k clauses of three literals we construct a set P of points. We show that there is a value K such that B is satisfiable if and only if $St\text{-}\Delta_2(P) = K$.

In Figure 6 a schematic layout of P is given for $B(x_0, x_1, x_2) = (x_0 \vee x_1 \vee \bar{x}_2) \wedge (x_0 \vee \bar{x}_1 \vee x_2) \wedge (\bar{x}_0 \vee \bar{x}_1 \vee \bar{x}_2)$ with $K = 39$. This is a grid of points with some well-defined holes. The maximum number of points in a horizontal or vertical line is K, and many lines have exactly K points. By Lemma 8 a full row or column in this setting has exactly $2K-2$ triangles. Intuitively, in this layout gadgets are separated by full rows and columns.

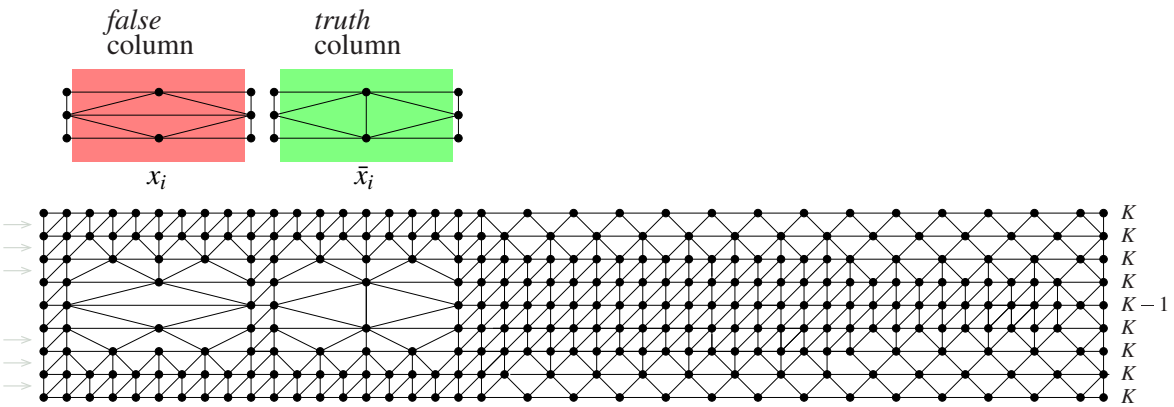

Figure 10: A variable gadget and how it is embedded in a grid of points. Arrows indicate full rows.

Figure 10 shows two horizontally aligned rectangles of eight points each that together represent a variable x_i. We call the collection of vertical lines that stab the left rectangle the x_i *column*, and the collection of vertical lines that stab the right rectangle the $\bar{x}_i$ *column* of the variable. Its functioning is essentially the same as that of the variable in Figure 8. We first assume that each rectangle full rows and columns as neighbors. We indicate how this can be achieved in horizontal direction in Figure 10. By Lemma 8 we conclude that all edges of the convex hull of each rectangle are present in any triangulation of minimal crossing number. The horizontal lines that contain the top and bottom lines of the two rectangles, respectively, contain K points each; the horizontal line that passes through the four middle points of the rectangles contains only $K-1$ points. Therefore, exactly one edge along this horizontal middle line may be missing. Because there is no point missing along the horizontal middle line except within the rectangles, the missing edge is in the interior of one of the two rectangles of the variable. We call the space that is spanned by all vertical lines that stab the rectangle with the missing edge the *true* column of the variable x_i. The space that is spanned by all vertical lines that stab the rectangle for which the middle edge is present is the *false* column of the variable x_i. Note that the *truth* column adds exactly one less to the vertical crossing number than the *false* column. In the overall layout, variable x_i is placed below and to the left of variable x_j for $i < j$ in such a way that variables are vertically separated from one another by a full row, and horizontally separated from one another by a full column.

Each literal is represented as a square of eight points. We make the rectangles of the variables wide enough to accommodate the necessary number of literals. That is, each rectangle of a variable is of a width twice the number of the maximum occurrence of a literal in B. Figure 10 gives a hint at how this widening of a rectangle is done technically. The three literals of a clause c_j are horizontally aligned, and the two rows that are spanned by them are called the clause c_j. If a literal x_i appears in the clause c_j, we place a literal gadget in the x_i column of the clause gadget c_j. If a literal $\bar{x}_i$ appears in the clause c_j, we place a literal gadget in the $\bar{x}_i$ column of the clause gadget c_j. The literal in column x_i of clause c_j is to the left of the literal in column x_i of clause c_h for $j < h$. Similarly, the literal in column $\bar{x}_i$ of clause c_j is to the left of the literal in column $\bar{x}_i$ of clause c_h for $j < h$. To make precise the relative position of the literals in a column we require that no vertical line stabs the interior of more than one literal.

By adding points to the right of the literals we ensure that a horizontal line through the top or bottom row of the three literals of a clause has exactly K points, and a horizontal line through the middle line of the three literals of a clause has exactly $K-2$ points. That is, two edges along these middle lines may be missing in a triangulation of minimum crossing number, but no more than two. As we will argue later, these edges will be missing in the interior of two of the literals. We call the missing of the horizontal middle edge in a literal the *false* setting of the literal, and the presence of this edge within a literal the *true* setting of the literal. Clauses are vertically separated from one another by a full row.

Finally, in adding points at the bottom of the clauses we ensure the following vertical point counts. First of all, the columns neighboring the variables have to be full, in particular, vertical lines that stab the left or the right points of a variable rectangle have K points each. A vertical line passing through the middle line of a variable rectangle has $K-1$ points. Together with the $K-1$ points on the horizontal middle line of each variable this implies that in any triangulation of minimum crossing number each variable will have exactly one *truth* and exactly one *false* column. A vertical line that passes through a vertical boundary of a literal, but not through a vertical boundary of a variable, receives $K-1$ points. A vertical line through the vertical middle line of some literal has only $K-3$ points. As Figure 6 suggests, when the distribution of points is done carefully—we give some details at the end of the proof—we end up with a configuration P of points that has a rectangular convex hull.

Let $B(x_0, x_1, \ldots, x_{n-1})$ be satisfiable. We show that P has a triangulation of crossing number $2K-2$ that is minimum by Lemma 8. All full rows and full columns are fully triangulated. If variable x_i has the value *true* we triangulate the interior of the two rectangles of variable x_i in such a way that the x_i column becomes this variable's *truth* column, and the $\bar{x}_i$ column becomes this variable's *false* column. The triangulation of the interior of the rectangles is reversed when variable x_i has the value *false*. We set each literal that represents the value *true* to its *true* setting, and set each literal that represents the value *false* to its *false* setting. The triangulation can be arbitrarily completed.

We can now convince ourselves that such a triangulation of P indeed attains $St\text{-}\Delta_2(P) = 2K-2$. No fully triangulated row or column has a crossing number larger than $2K-2$. Because exactly one edge is missing in the horizontal middle line of each respective variable the two rows of a variable are full. In each clause there is at least one literal in its *true* setting. Therefore, we can afford at most two extra triangles caused by the *false* setting of the other two literals in the clause, and no row of a clause intersects more than $2K-2$ triangles. Finally, we may have one edge missing on the vertical lines passing trough the middle of a literal only in a *truth* column. This condition holds by definition of the setting of the literals according to the truth value of the variables. The "arbitrary completion" of the triangulation only happens in the lower right of the construction. There, however, except for the boundary, vertical and horizontal lines have a low point count, and the allowed crossing number is not exceeded.

Assume now on the other hand that there is a triangulation of P that has crossing number $2K-2$. We show that B is satisfiable. Because Since a full row or column can be triangulated in such a way that the crossing number of $2K-2$ is not exceeded we only have to take care of the rows and columns were we have a degree of freedom, and where the point count is critical by Lemma 8. Because Since the points in the top and the middle row of a clause add up to $2K-2$, we can afford at most two literals of each clause set to *false*. One literal in each clause has to be set to *true*. The *true* literal has to be in a *true* column of a variable, for otherwise the vertical crossing number would exceed $2K-2$. Any other literal in this column can be

also set to *true*. The horizontal point count of the horizontal lines of the variable force the second column of this variable to be a *false* column. In order not to exceed the allowed crossing number in vertical direction, all literals in this column have to be set to *false*. As one easily checks, this yields a consistent setting of the variables, and B is satisfiable.

We conclude our proof by giving some more details about how the points in the grid P are distributed, thereby roughly estimating the size of K. The size of the construction is determined by the horizontal and vertical middle lines passing through variables. A variable is at most $2k$ points wide, i.e., the variable rectangles consume a total width of $n(2k+2)$ points. In order to achieve a point count of K in each horizontal line through a variable, and of $K-1$ for the middle line, we have to add points to the right of the rectangles.

Note that the grid can easily be turned into a square. Finally, it is straightforward, though tedious, to compute the exact numbers of triangles used in the construction, in particular in the "horizontal fill" right of the variables and the "vertical fill" below the clauses. Clearly, these numbers are polynomial in n and k. □

Proof of Lemma 10:

Among the fractional perfect matchings of minimum stabbing number, consider a solution x that minimizes the total stabbing number, i.e., the sum of stabbing numbers over all combinatorial lines. We claim that the support graph of x cannot contain any crossing pair of edges.

Suppose there were $e_{13} := \{v_1, v_3\}$, $e_{24} := \{v_2, v_4\}$ with $x(e_{13}) \geq 0$ and $x(e_{24}) \geq 0$, such that e_{13} and e_{24} cross. Consider $e_{12} := \{v_1, v_2\}$, $e_{34} := \{v_3, v_4\}$, $e_{14} := \{v_1, v_4\}$, $e_{23} := \{v_2, v_3\}$, and a sufficiently small $\varepsilon \leq \min\{x_{e_{13}}, x_{e_{24}}\}$. As $\sum_{e \in \delta(v_i)} x_e = 1$ and $x(e_{13}) > 0$, $x(e_{24}) > 0$, we have $x(e_{12}) < 1-\varepsilon$, $x(e_{34}) < 1-\varepsilon$, $x(e_{14}) < 1-\varepsilon$, $x(e_{23}) < 1-\varepsilon$.

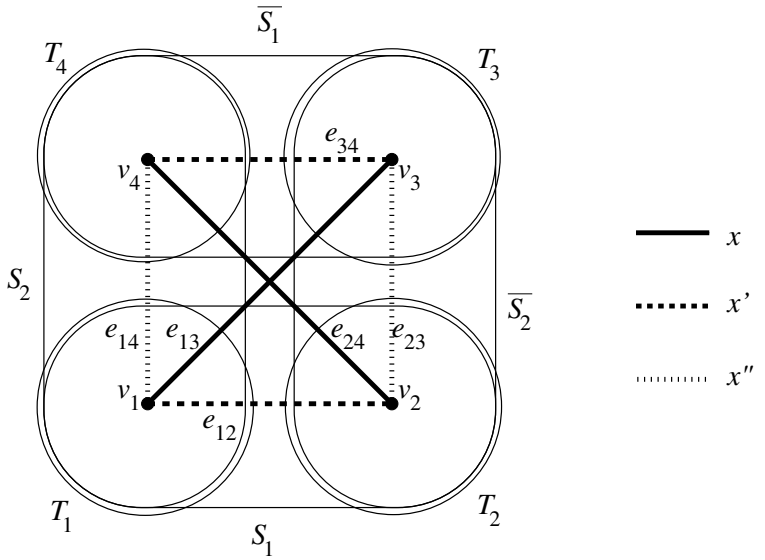

Figure 11: Uncrossing a fractional solution while preserving all blossom inequalities.

Consider two possible alternative solutions arising from "uncrossing" e_{13} and e_{24}. Let x' and x'' be defined by

$$x'_e := \begin{cases} x_e - \varepsilon & \text{for} \quad e = e_{13}, e_{24}, \\ x_e + \varepsilon & \text{for} \quad e = e_{12}, e_{34}, \\ x_e & \text{for} \quad \text{all other } e, \end{cases} \tag{10}$$

and

$$x''_e := \begin{cases} x_e - \varepsilon & \text{for} \quad e = e_{13}, e_{24}, \\ x_e + \varepsilon & \text{for} \quad e = e_{14}, e_{23}, \\ x_e & \text{for} \quad \text{all other } e. \end{cases} \tag{11}$$

By convexity, both x' and x'' satisfy all stabbing constraints that are valid for x. Furthermore, it is easy to see that both x' and x'' have shorter total stabbing number than x. Thus, we need to argue that at least one of these solutions satisfies all blossom inequalities.

Assume that for each of the two alternative solutions, there is a violated blossom inequality, i.e., there are two odd sets, $S_1 \subset V$ and $S_2 \subset V$, such that $\sum_{e \in \delta(S_1)} x'(e) < 1$ and $\sum_{e \in \delta(S_2)} x''(e) < 1$, while $\sum_{e \in \delta(S_1)} x(e) = 1$ and $\sum_{e \in \delta(S_2)} x(e) = 1$. Let $s_i := |\{v_1, v_2, v_3, v_4\} \cap S_i|$. It is straightforward to see that for $s_i \in \{0, 1, 3, 4\}$, both x' and x'' satisfy all blossom inequalities that are valid for x. Therefore, we only need to consider $v_1, v_2 \in S_1$ and $v_3, v_4 \in \overline{S_1}$, and $v_1, v_4 \in S_2$ and $v_2, v_3 \in \overline{S_2}$. Let $T_1 := S_1 \cap S_2$, $T_2 := \overline{S_1} \cap S_2$, $T_3 := S_1 \cap \overline{S_2}$, $T_4 := \overline{S_1} \cap \overline{S_2}$ and $E_{ij} := \{e = \{x, y\} \in E \mid x \in T_i, y \in T_j\}$. As $|T_1 \cup T_2|$, $|T_1 \cup T_4|$, $|T_2 \cup T_3|$, and $|T_3 \cup T_4|$ are all odd, we may assume without loss of generality that $|T_1|$ and $|T_3|$ are odd, and $|T_2|$ and $|T_4|$ are even. Then we have

$$\begin{aligned} 2 &= \sum_{e \in \delta(S_1)} x(e) + \sum_{e \in \delta(S_2)} x(e) \\ &= \sum_{e \in \delta(T_1)} x(e) + \sum_{e \in \delta(T_3)} x(e) + 2 \sum_{e \in E_{24}} x(e) \\ &\geq \sum_{e \in \delta(T_1)} x(e) + \sum_{e \in \delta(T_3)} x(e) + 2x(e_{24}) \\ &> \sum_{e \in \delta(T_1)} x(e) + \sum_{e \in \delta(T_3)} x(e). \end{aligned}$$

This implies that $\min\{\sum_{e \in \delta(T_1)} x(e), \sum_{e \in \delta(T_3)} x(e)\} < 1$, contradicting the assumption that x satisfies all blossom inequalities.

□

Proof of Lemma 11:

We proceed completely analogous to the proof of the previous lemma to deduce that we can perform an uncrossing 2-exchange. In addition, note that constraints (7) imply that in the presence of two crossing edges in the support graph, no two adjacent ones of the other four edges $\{v_1, v_2\}$, $\{v_2, v_3\}$, $\{v_3, v_4\}$, $\{v_4, v_1\}$ may both have weight 1.

□

Appendix B: Benchmark Instances for *St-Mat*$_2(P)$

We use the commercial LP/IP solver CPLEX 8.0. Table entries bearing the † sign indicate an exceeding of our CPU time limit of four hours for solving the IP, i.e., computing the exact optimum. For the results listed below we did not make use of any blossom inequalities in the LPs, as results are already excellent. We also did some testing based on available separation routines for blossoms.

Below we give detailed give results for TSPLIB and clustered instances (Table 1) and for grid and random instances (Table 2). Each table lists the instance name (which indicates the number of points), the value of the optimal lower provided by the LP, the CPU time in seconds on a Pentium III 700 MHz PC with 1GB main memory running Linux, the IP optimal value, the CPU time to solve the IP, and the stabbing numbers obtained from iterated rounding.

Instance	LP opt	LP CPU	IP opt	IP CPU	iter. rounding
ulysses22	1.992308	0.01	2	0.05	2
berlin52	2.815158	0.52	4	8.86	5
lin105	5.500000	1.21	infeas.	†	7
bier127	4.330856	5.64	infeas.	†	5
u159	15.000000	1.76	infeas.	†	15
ts225	13.750000	5.37	infeas.	†	16
tsp225	11.500000	62.06	infeas.	†	12
a280	10.500000	55.12	$\leq$12	†	13
lin318	8.113143	131.54	$\leq$12	†	11
pcb442	16.500000	270.03	$\leq$19	†	18
c101	7.000000	0.10	7	1.75	8
c102	7.000000	0.10	7	1.75	8
c103	7.000000	0.10	7	1.75	8
c104	7.000000	0.10	7	1.75	8
c105	7.000000	0.10	7	1.75	8
c106	7.000000	0.10	7	1.75	8
c107	7.000000	0.10	7	1.75	8
c108	7.000000	0.10	7	1.75	8
c201	6.000000	0.50	6	2.00	7
c202	6.000000	0.45	6	2.00	7
c203	6.000000	0.46	6	1.95	7
c204	6.000000	0.46	6	1.95	7
c205	6.000000	0.46	6	1.95	7
c206	6.000000	0.46	6	1.92	7
c207	6.000000	0.46	6	1.95	7
c208	2.953237	0.58		†	5

Table 1: Comparison of bounds for *St-Mat*$_2(P)$ and iterated rounding: TSPLIB and clustered instances.

Figures 12, 13 14, 15 show several fractional LP solutions. Edge weight is represented by thickness of edges. The instances in Figures 14 and 15 were not re-solved with respect to total stabbing depth; the point is to demonstrate that achieving planarity is not necessary from a practical point of view: The resulting graphs are still rather sparse, showing that re-solving the LP is in general not necessary in order to obtain heavy edges. This saves running time, as each LP needs to be solved only once instead of twice.

Instance	LP opt	LP CPU	IP opt	IP CPU	iter. rounding
grid5a	2.500000	0.00	3	0.01	3
grid5b	2.750000	0.00	3	0.02	3
grid5c	2.750000	0.01	3	0.02	4
grid5d	2.000000	0.00	3	0.03	3
grid5e	2.500000	0.00	3	0.02	3
grid8a	5.003205	0.04	6	0.17	6
grid8b	5.125000	0.05	6	0.21	6
grid8c	5.000000	0.04	5	0.12	6
grid8d	5.428571	0.04	6	0.19	7
grid8e	5.403226	0.04	6	0.29	6
grid10a	4.250000	0.12	5	0.68	6
grid10b	4.250000	0.10	5	0.40	6
grid10c	5.250000	0.11	6	245.28	6
grid10d	4.500000	0.07	5	13.58	6
grid10e	5.000000	0.03	5	1.36	6
grid15a	6.000000	0.73	6	8.40	7
grid15b	7.500000	0.47	8	9.90	8
grid15c	6.000000	1.00	6	3.27	6
grid15d	6.500000	0.24	7	†	7
grid15e	6.750000	0.74	7	3.91	7
grid20a	9.166667	15.38	$\leq$11	†	12
grid20b	9.250000	20.99	$\leq$11	†	11
grid20c	9.500000	8.70	$\leq$11	†	11
grid20d	9.500000	26.69	$\leq$11	†	11
grid20e	10.000000	20.43	$\leq$11	†	12
random10a	1.750000	0.00	2	0.01	2
random10b	1.833333	0.00	2	0.00	2
random10c	1.750000	0.00	2	0.01	2
random10d	1.700000	0.00	2	0.01	2
random10e	1.812500	0.00	2	0.01	2
random50a	2.594823	0.25	3	19.83	5
random50b	2.628112	0.23	3	1.91	4
random50c	2.668918	0.23	4	30.77	4
random50d	2.661581	0.22	4	15.99	4
random50e	2.789609	0.33	4	25.54	5
random100a	3.376247	5.57		†	6

Table 2: Comparison of the LP/IP bounds for $St\text{-}Mat_2(P)$ and iterated rounding: grid and random instances.

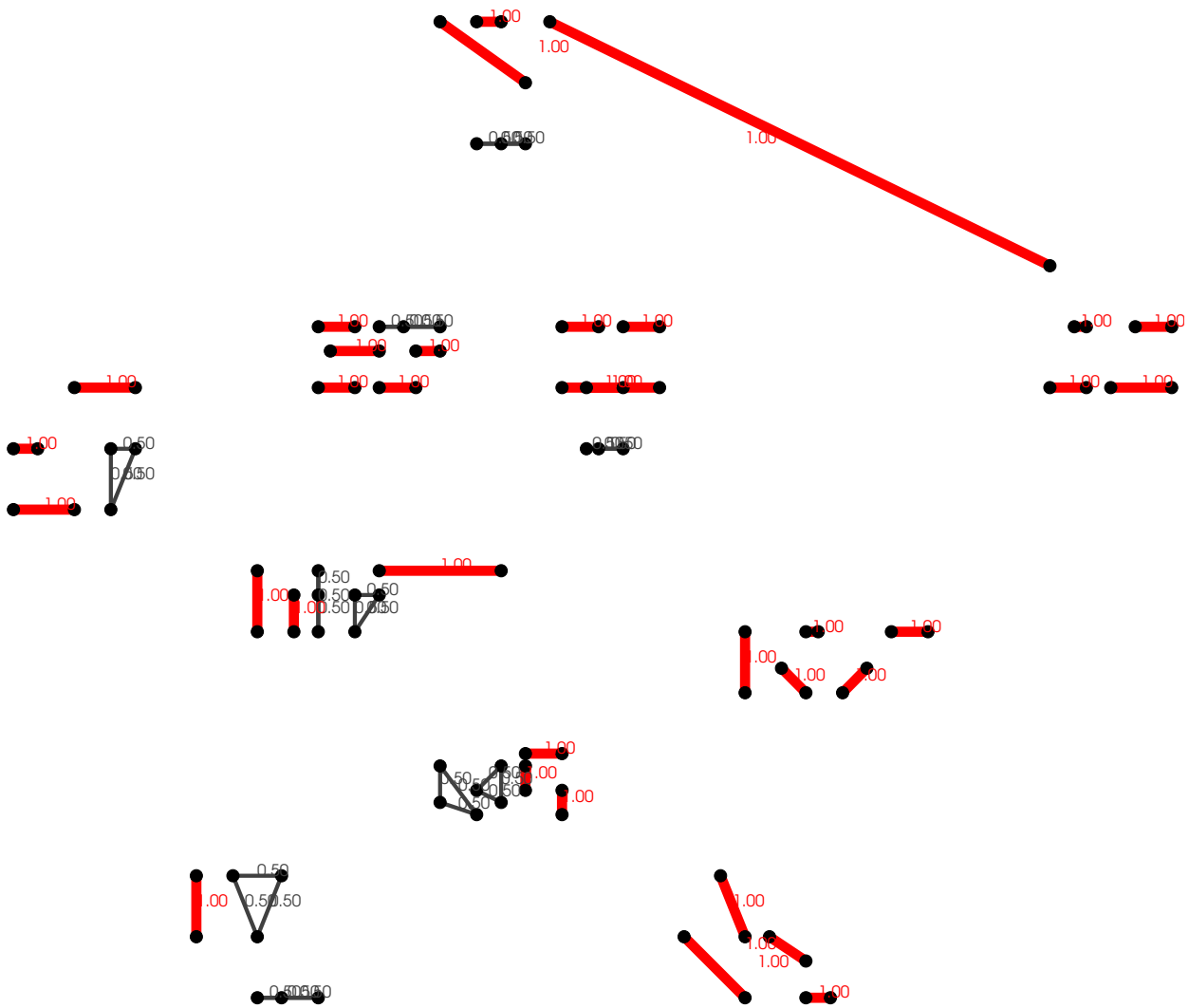

Figure 12: LP optimal solution for c101

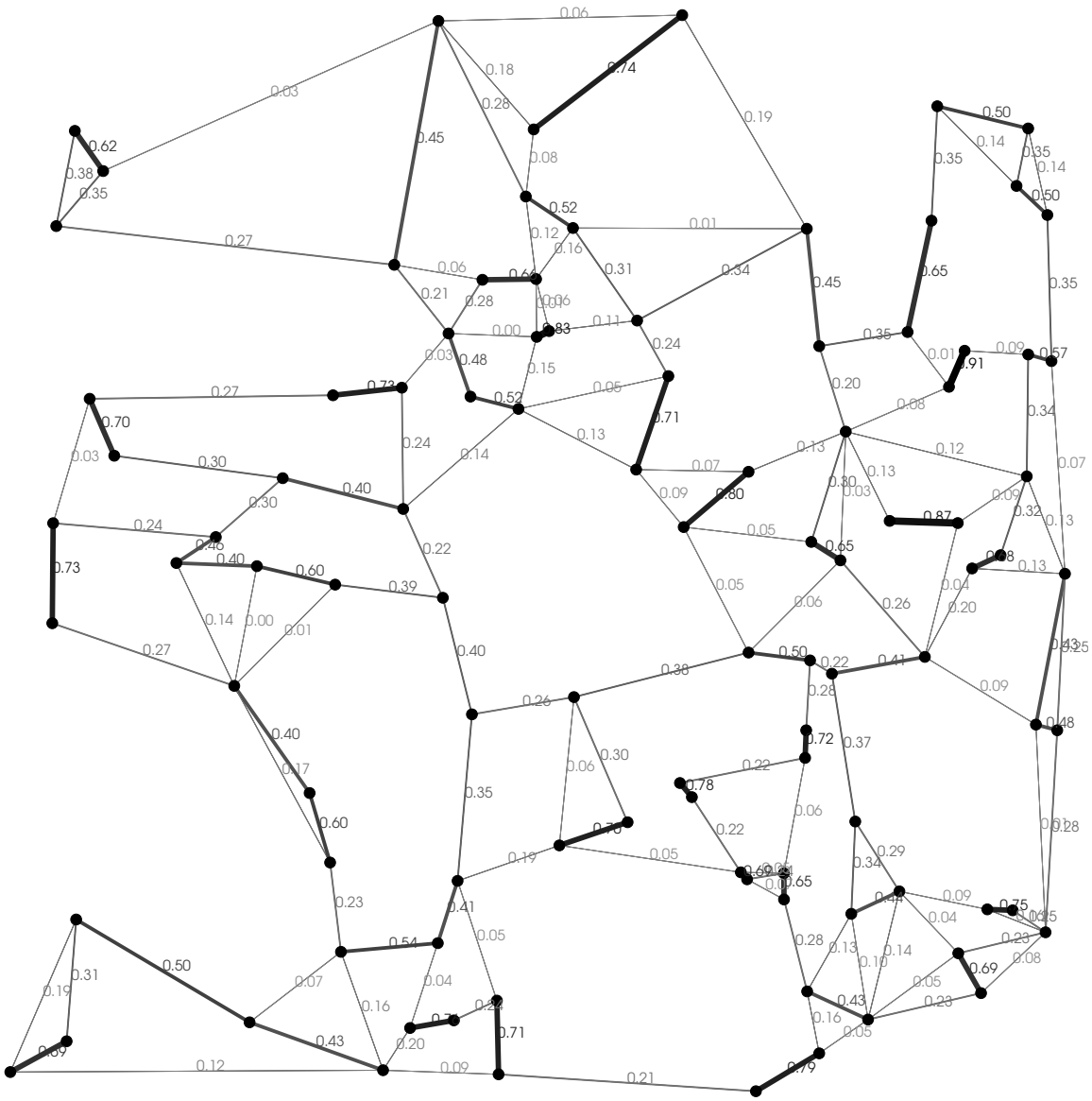

Figure 13: LP optimal solution for random100a

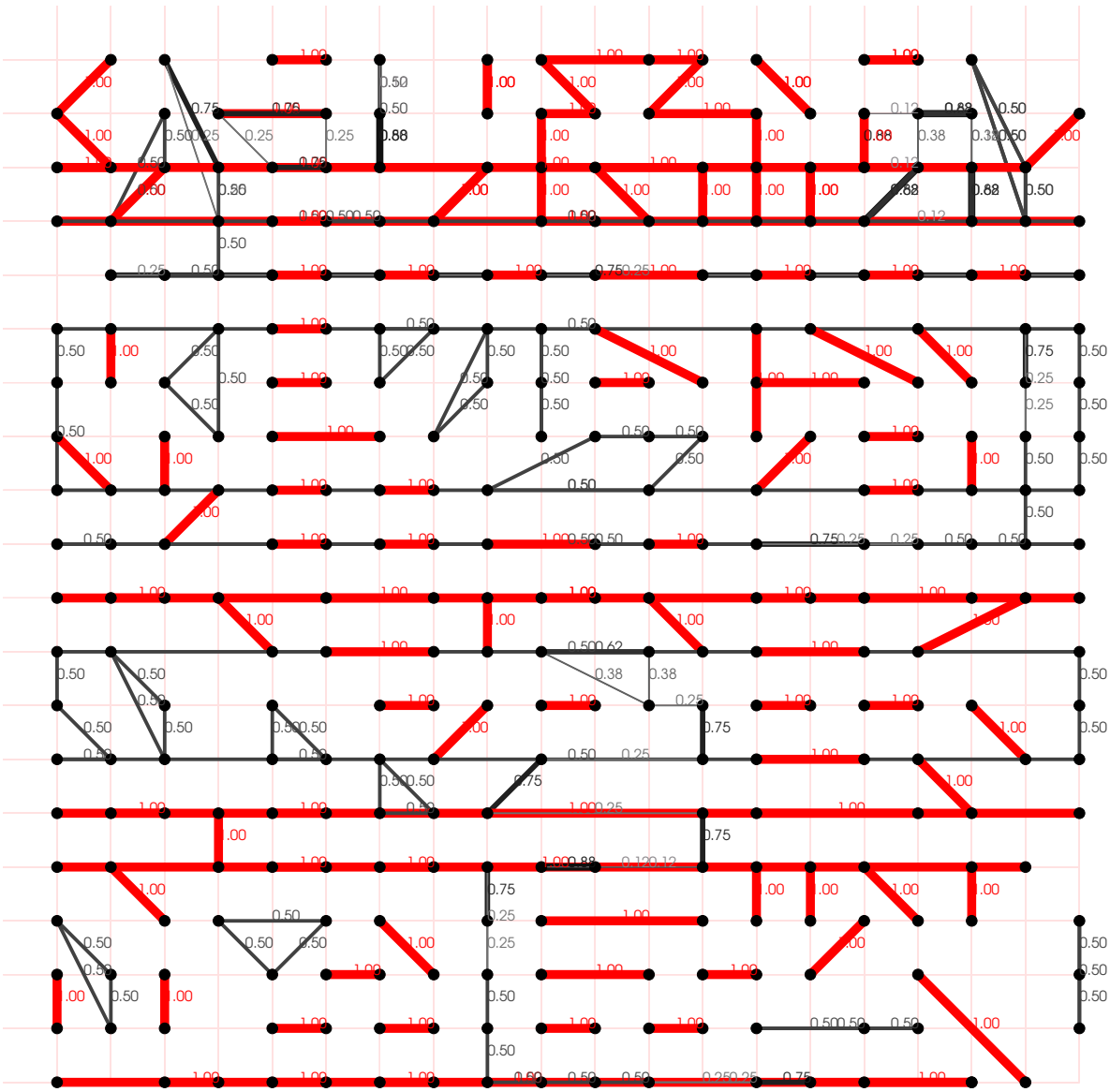

Figure 14: LP optimal solution for grid20b

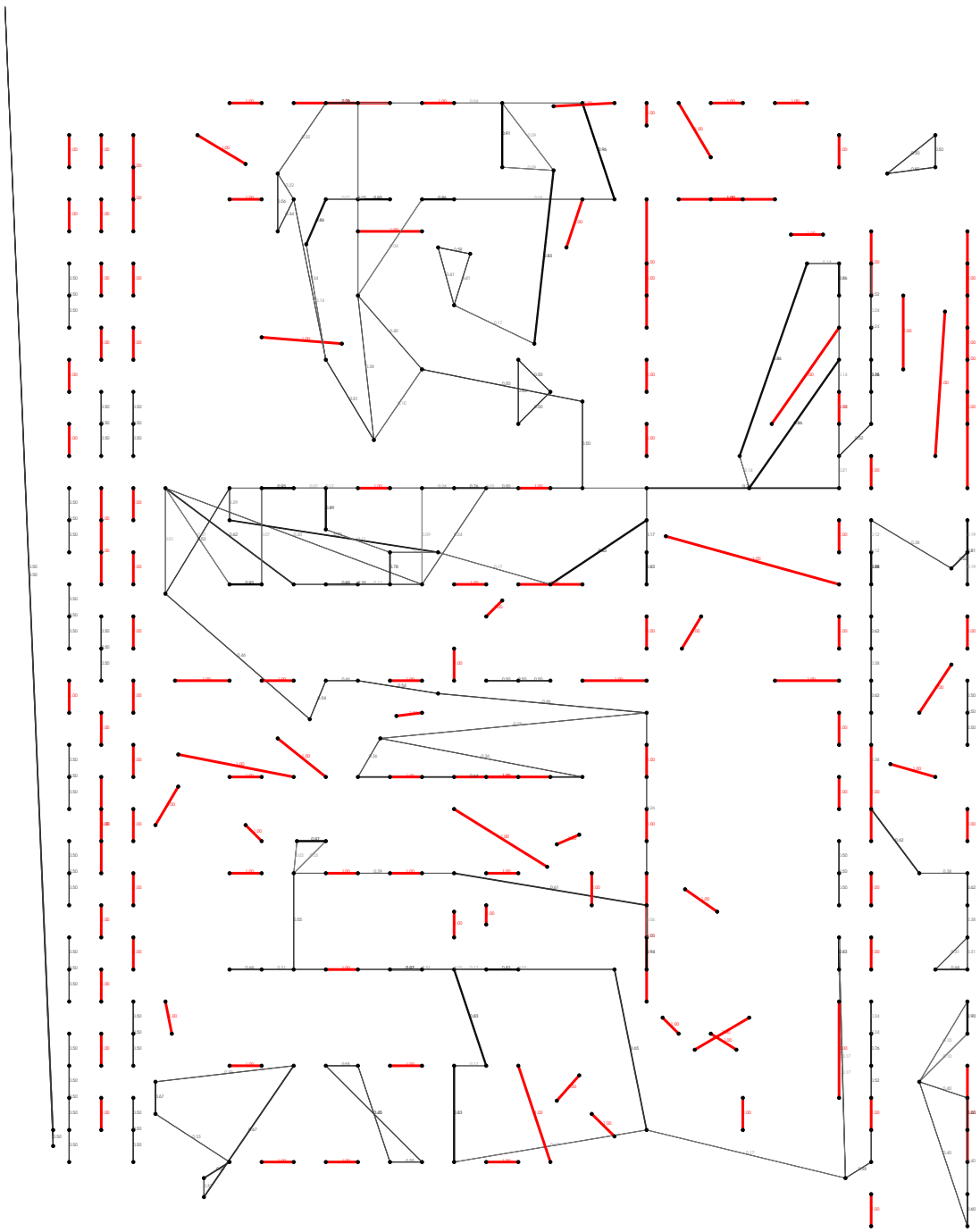

Figure 15: LP optimal solution for pcb442