Spanning Trees with Few Crossings in Geometric and Topological Graphs

Christian Knauer* Étienne Schramm† Andreas Spillner‡ Alexander Wolff†

Abstract

We study the problem of computing a spanning tree in a connected topological graph such that the number of crossings in the spanning tree is minimum. We show it is NP-hard to find a good approximation of the minimum number of crossings even in geometric graphs. On the other hand we show that the problem is fixed-parameter tractable and present a mixed-integer linear program formulation.

1 Introduction

Let $G(V, E)$ be an undirected graph that is embedded in the plane, so that no two edges share an unbounded number of points. We call G a *topological graph*. If all edges are straight-line embedded, then G is called a *geometric* graph.

A *crossing* $\{e, e'\}$ is a pair of edges such that $e \cap e' \not\subseteq V$. We call $\mu_{ee'} = |(e \cap e') \setminus V|$ the *multiplicity* of the crossing $\{e, e'\}$. Let $X \subseteq \binom{E}{2}$ be the set of pairs of crossings in E. Note that c edges intersecting in a single non-endpoint give rise to $\binom{c}{2}$ crossings. We will use n, m, and k as shorthand for the cardinalities of V, E, and X, respectively. We will use the notation $G(V, E, X)$ if we want to emphasize the connection between G and X. We define the *weighted number of crossings* of a subgraph $G(V, E', X')$ of G as $\sum_{\{e,e'\} \in X'} \mu_{ee'}$.

In this paper we study the problem of computing a spanning tree of a given connected topological graph such that the weigted number of crossings in the tree is minimum.

Kratochvil et al. [2] have shown that for topological graphs it is NP-hard to decide whether they contain crossing-free subgraphs of certain types, such as cycles, u–v paths, maximum matchings, or spanning trees. Jansen and Woeginger [1] have strengthened the last result by showing that it is even NP-hard to decide whether or not a *geometric* graph has a crossing-free spanning tree.

*Institute of Computer Science, Freie Universität Berlin, Germany, christian.knauer@inf.fu-berlin.de

†Fakultät für Informatik Universität Karlsruhe, Germany, http://i11www.ira.uka.de/people. Supported by grant WO 758/4-1 of the German Science Foundation (DFG).

‡Institute of Computer Science, Universität Jena, Germany, spillner@minet.uni-jena.de

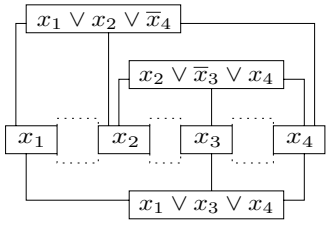

Figure 1: Overall structure of G_F.

That result does not rule out the existance of efficient constant-factor approximation algorithms. However, as we will show in Section 2, such algorithms do not exist unless $\mathcal{P} = \mathcal{NP}$. So we turned our attention to other possible ways to attack the problem: in Section 3 we show that the problem under consideration is fixed-parameter tractable with k being the parameter and in Section 4 we present a mixed-integer linear program (MIP) formulation. We conclude in Section 5.

2 Hardness of approximation

Let $\phi(G)$ be 1 plus the minimum number of crossings in a spanning tree of a geometric graph G. The main result of this section is the following theorem.

Theorem 1 *Given geometric graph G, it is NP-hard to approximate $\phi(G)$ within a factor of $k^{1-\varepsilon}$ for any $\varepsilon > 0$.*

Note that it is trivial to give a factor-$(k+1)$-approximation and since a geometric graph is also a topological graph, theorem 1 obviously remains valid for topological graphs.

We obtain this result by a reduction from planar 3SAT [3]. Our reduction maps a given planar 3SAT formula F to a geometric graph G_F such that G_F has a plane spanning tree iff F has a fulfilling truth assignment. The overall structure of G_F is indicated in Figure 1 for $F = (x_1 \vee x_2 \vee \overline{x}_4) \wedge (x_2 \vee \overline{x}_3 \vee x_4) \wedge (x_1 \vee x_3 \vee x_4)$.

We have a gadget for every variable occuring in F. These gadgets are arranged along a horizontal line ℓ. We further have a gadget for every clause in F which is connected with every variable occuring in the clause. This gadget looks like a three-legged comb.

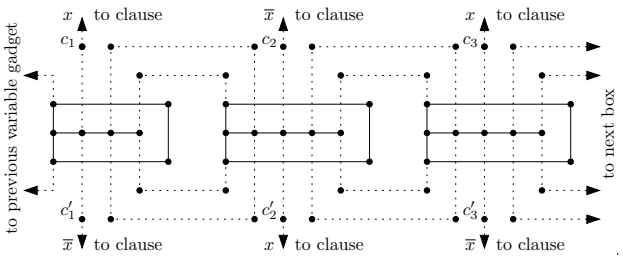

Figure 2: Part of a variable gadget.

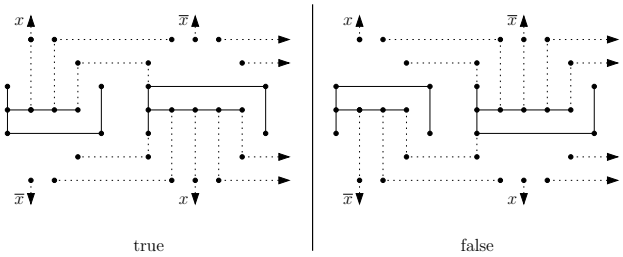

Figure 3: Spanning trees encoding true and false.

Now let's have a closer look at the gadgets. The leftmost part of the gadget for a variable x is shown in Figure 2. The gadget for x consists of at most twice as many boxes as there are clauses in F that contain x. Three of these boxes are drawn with solid edges in Figure 2. The dotted edges that emanate from the boxes fulfill three tasks. First they connect consecutive boxes within one variable gadget. Second they connect the first and last box of variable gadgets that are consecutive on the line ℓ. Third they connect boxes to clause gadgets. Each dotted edge that connects a variable gadget to a clause gadget is associated with a literal. This literal will be true if the dotted edge is part of the spanning tree of G_F.

The intended way of simulating the truth assignment of the variable x is indicated in Figure 3. The Boolean values of x correspond to the two ways in which a crossing-free spanning tree can be chosen among the edges of the gadget of x. Note that only every other box can be connected to a clause gadget above (below) ℓ. This way we ensure that according to the value of x either only the dotted edges associated to positive literals or only the dotted edges associated to negative literals can connect x to clause gadgets. Not all points of type c_i or c'_i in Figure 2 are used—only those where the variable is in fact connected to a clause gadget in G_F. For example, the gadget for x_2 in Figure 1 consists of three boxes, but only the points c_1 and c_3 are used.

A clause gadget is shown in Figure 4(a). The three boxes in the lower part of the figure belong to the gadgets of those variables that form the clause. Note that the clause gadget is connected to the rest of G_F only via the dotted edges that go into these three

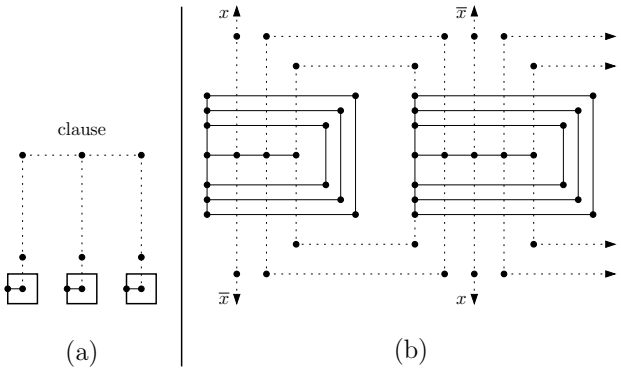

Figure 4: Clause gadget and modified variable gadget.

boxes. Thus G_F has a crossing-free spanning tree only if at least one of the three literals in the clause is true.

Up to this point our construction only proves that deciding whether or not a geometric graph has a crossing-free spanning tree is NP-hard. However, our construction is designed to allow us to apply a simple trick that yields the desired hardness of approximation. We substitute certain solid edges in every box in every variable gadget with z new edges. This is indicated for $z = 3$ in Figure 4(b).

If we set z to a polynomial in the length of F with large enough degree, our reduction remains polynomial and Theorem 1 is an immediate consequence of the following lemma.

Lemma 2 *If F has a satisfying truth assignment then $\phi(G_F) = 1$, else $\phi(G_F) \geq z$.*

Proof. First suppose F has a satisfying truth assignment. Then we select edges for a spanning tree of G_F in every variable gadget as indicated in Figure 3 according to the satisfying truth assignment. Since in every clause of F at least one literal is true under the satisfying truth assignment it is possible to connect every clause gadget in a spanning tree without any crossings. Thus $\phi(G_F) = 1$.

Now suppose F does not have a satisfying truth assignment. Consider a box B after substitution of edges as in Figure 4(b). Then B has z solid horizontal edges above (below) ℓ and at most four dotted vertical edges above (below) ℓ that intersect the former. Let T be a spanning tree of G_F with the minimum number of crossings. Let v_{top} (v_{bot}) be the number of vertical edges in T that are above (below) ℓ and intersect B. Similarly, let h_{top} (h_{bot}) be the number of horizontal edges in T that are above (below) ℓ and belong to B. Since the vertices on the right of the box can only be connected to the rest of the graph via the solid top or bottom edges, we have $h_{\text{top}} + h_{\text{bot}} \geq z$. Now suppose $v_{\text{top}} \leq v_{\text{bot}}$. Then there are $v_{\text{top}}h_{\text{top}} + v_{\text{bot}}h_{\text{bot}} \geq v_{\text{top}}z$ crossings. If we remove all solid bottom edges of B from T and instead use all solid top edges and

all solid vertical edges of B, we get a spanning tree T' which does not have more crossings than T. So we may suppose that in every box of G_F all the solid top edges or all the solid bottom edges are in T. We distinguish two cases.

Case 1: There are two consecutive boxes in a variable gadget such that from both boxes all the solid top edges (all the solid bottom edges) are in T. Then there are at least z crossings in T between the solid top edges in the two boxes and the dotted edges that connect these two boxes. Thus $\phi(G_F) \geq z$.

Case 2: There are no two consecutive boxes in a variable gadget such that from both boxes all the solid top edges (all the solid bottom edges) are in T. Then according to Figure 3 we derive a truth assignment β from the structure of T inside every variable gadget. Since F does not have a satisfying truth assignment there is a clause C in F which is false under β. Consider the clause gadget associated with C. This clause gadget must be connected in T with the rest of G_F which leads to at least z crossings. Thus again $\phi(G_F) \geq z$. □

Instead of searching for a spanning tree with a minimum number of crossings one could also ask for a crossing-free spanning forest of a geometric graph G with the minimum number $\phi'(G)$ of trees. But using a similar reduction as the one sketched above we obtain the following theorem.

Theorem 3 *It is NP-hard to approximate $\phi'(G)$ within a factor of $n^{1-\varepsilon}$ for any $\varepsilon > 0$.*

3 Fixed-parameter tractability

We first show that deciding whether or not a topological graph G has a crossing-free spanning tree is fixed-parameter tractable where we use the total number of crossings k as fixed parameter. We set $E_X = \bigcup X$. Thus E_X contains exactly those edges that participate in a crossing. Note that $|E_X| \leq 2k$ and observe that G has a crossing-free spanning tree iff G has a crossing-free connected spanning subgraph. Thus it is sufficient to examine all maximal crossing-free subgraphs of G and check whether one of them is connected. To this end we proceed as follows:

1. Remove all edges in E_X from G. The resulting graph $G' = (V, E')$ is crossing-free.
2. For all maximal crossing-free subsets $H \subseteq E_X$ check whether the graph $G' \cup H = (V, E' \cup H)$ is connected.

Note that in the second step we can actually shrink each connected component of G' to a single vertex. If G' has more than $k+1$ connected components, adding the at most k edges in H cannot make G' connected. Therefore we only have to consider a graph with $O(k)$ vertices.

It can easily be shown by induction on k that there are at most 2^k maximal crossing-free subsets H. Since detecting all the crossings in G can be done in $O(k + m \log m)$ time and generating and checking a subset H can be done in $O(k)$ time, we have the following theorem.

Theorem 4 *We can decide in $O(m \log m + k2^k)$ time whether a topological graph has a crossing-free spanning tree.*

Checking not only the crossing-free subsets of E_X but all subsets we obtain an algorithm that finds a spanning tree with the minimum weighted number of crossings.

Theorem 5 *Given a topological graph we can compute in $O(m \log m + k4^k)$ time a spanning tree with the minimum weighted number of crossings.*

In the remainder of this section we sketch how to speed up the decision algorithm in the special case that the crossings in X are pairwise disjoint.

Suppose G' has at most $k+1$ connected components and we have shrunken each to a single vertex. Our new algorithm distinguishes two cases. Let $\alpha = \frac{4}{5}$ and r be the number of connected components of G'.

Case 1: $r \leq \alpha k$. If G has a connected crossing-free spanning subgraph at all, then there are at least $2^{(1-\alpha)k}$ crossing-free subsets $H \subseteq E_X$ such that $G' \cup H$ is connected. This can be seen as follows: Suppose there is a crossing-free subset $F \subseteq E_X$ that makes G' connected. Then we can choose such an F with $|F| \leq r - 1 < \alpha k$. Let $X_F = \{c \in X \mid c \cap F = \emptyset\}$. Since the elements of X are pairwise disjoint we have $|X_F| > (1 - \alpha)k$. If we select one edge from each crossing in X_F and add these edges to F the resulting set of edges is still crossing-free. There are at least $2^{(1-\alpha)k}$ possible ways to select edges. Thus there are at least $2^{(1-\alpha)k}$ crossing-free subsets $H \subseteq E_X$ such that $G' \cup H$ is connected.

This suggests the following randomized algorithm: From each element of X we randomly select one edge and check if the resulting crossing-free graph is connected. If the given geometric graph G has a crossing-free connected spanning subgraph, the probability of success is at least $2^{(1-\alpha)k}/2^k = 2^{-\alpha k}$. Thus $O(2^{\alpha k})$ iterations suffice with high probability and the expected running time is $O(m \log m + k2^{\alpha k})$ with high probability.

Case 2: $r > \alpha k$. We define the degree of a connected component C of G' as the number of edges in E_X that are incident to a vertex that belongs to C.

We observe that for each component of degree one there is exactly one edge in E_X to connect this com-

ponent to the rest of G'. Thus this edge must be selected for any connected spanning subgraph of G.

Furthermore we can assume that for every crossing $c \in X$ none of the two edges in c connects vertices in the same component of G', since such an edge could be deleted which would also make the crossing c vanish.

Next we argue that if $r > \alpha k$ there is always at least one component of G' that has degree at most 4. For if all components had degree at least 5, we would get $4k \geq 2|E_X| \geq 5r > 5\alpha k = 4k$.

Thus our algorithm selects a component C of G' such that the degree d of C is at most 4. The key observation is that one of the d edges in E_X incident to a vertex of C must be chosen to connect C to the rest of G'. This means that we do not have to try all 2^d subsets of those d edges but only $2^d - 1$. This observation can be made more precise by a detailed case analysis which we omit in this abstract. The case analysis yields the following recurrence for the running time $T(k)$ of our algorithm:

$$T(k) \leq 15T(k-4) + O(1)$$

This solves to $T(k) \in O(15^{k/4})$. To summerize we state the following theorem.

Theorem 6 *There is a Monte Carlo algorithm with one sided error and expected running time in $O(m \log m + k1.968^k)$ which decides whether a given geometric graph with pairwise disjoint crossings has a crossing-free spanning tree.*

4 A Mixed-Integer Linear Program

In this section we give a MIP formulation for the problem of computing a connected spanning subgraph G' with the minimum weigted number of crossings for a given topological graph $G(V, E)$. We introduce a variable $x_{ee'}$ that is designed to punish a crossing between the edges e and e'.

Our objective function is

$$\min \sum_{\{e,e'\} \in X} \mu_{ee'}\, x_{ee'}. \qquad (1)$$

We have the following constraints. We introduce a variable y_e for each edge $e \in E$ that is 1 iff e will be part of G'. For each pair $\{e, e'\} \in X$ we require:

$$x_{ee'} \geq 0 \quad \text{and} \quad x_{ee'} \geq y_e + y_{e'} - 1 \qquad (2)$$

Given our objective function (1) and the fact that y_e will be forced to lie in $\{0, 1\}$, Constraint (2) is equivalent to $x_{ee'} = \min\{y_e, y_{e'}\}$. In other words if we manage to make sure that the graph $G' = (V, E')$ with $E' = \{e \in E \mid y_e = 1\}$ is connected, then the variables of type $x_{ee'}$ in fact count the number of crossings in G'.

We model connectivity by fixing an arbitrary vertex $s \in V$ as sink and then introducing flow between s and each other vertex $t \in V \setminus \{s\}$. The flow is modeled by a 0–1 variable f_e^t for each edge $e \in E$. First we make sure that for each choice of t, the source s and the sink t have exactly one edge with flow:

$$\sum_{e \text{ incident to } s} f_e^t \;=\; \sum_{e \text{ incident to } t} f_e^t \;=\; 1 \qquad (3)$$

Note that if a graph contains an s–t path, then that graph also contains an s–t path that visits each vertex at most once. So we simply ensure that each vertex $v \in V \setminus \{s, t\}$ has either zero or two incident edges with flow. To do this we need an auxiliary 0–1 variable h_v^t for each v. Now we can model our special kind of flow conservation in each vertex $v \in V \setminus \{s, t\}$.

$$\sum_{e \text{ incident to } v} f_e^t \;=\; 2h_v^t \qquad (4)$$

Finally, for each edge $e \in E$ we lower-bound the "global" decision variable y_e (that decides whether e goes into the spanning subgraph G') by the "local" flow f_e^t (that goes through e from s to t):

$$y_e \geq f_e^t \qquad (5)$$

Given our objective function and Constraint (2), this is equivalent to setting $y_e = \max\{f_e^t \mid t \in V \setminus \{s\}\}$. This completes our MIP formulation. It consists of $O(nm + k)$ variables and constraints.

5 Conclusion

We have shown that attacking the problem of finding a spanning tree with few crossings in a given topological graph by approximation algorithms is probably not a good idea, even though it could be an interesting theoretical problem to find approximation algorithms with approximation factor in $o(k)$.

On the other hand we have presented a fixed-parameter algorithm where we used the total number of crossings as fixed parameter. There might be other quantities associated with a geometric graph that could be used as fixed parameter. We are currently investigating this.

Finally we have given a MIP formulation, which we plan to implement in order to evaluate several heuristics.

References

[1] K. Jansen and G. J. Woeginger. The complexity of detecting crossingfree configurations in the plane. *BIT*, 33:580–595, 1993.

[2] J. Kratochvil, A. Lubiw, and J. Nesetril. Noncrossing subgraphs in topological layouts. *SIAM J. Disc. Math.*, 4(2):223–244, 1991.

[3] D. Lichtenstein. Planar formulae and their uses. *SIAM J. Computing*, 11:329–343, 1982.