Improved Fixed-Parameter Algorithms for Non-Crossing Subgraphs[1]

Magnús M. Halldórsson
Dept. of Computer Science, Faculty of Engineering, University of Iceland, IS-107 Reykjavik, Iceland. mmh@hi.is

Takeshi Tokuyama[2]
Graduate School of Information Sciences, Tohoku University, Sendai, 980-8579 Japan.
tokuyama@dais.is.tohoku.ac.jp

Alexander Wolff[3]
Fakultät für Informatik, Universität Karlsruhe, P.O. Box 6980, 76128 Karlsruhe, Germany.
http://i11www.ira.uka.de/people/awolff

Abstract

We consider the problem of computing non-crossing spanning trees in topological graphs. It is known that it is NP-hard to decide whether a topological graph has a non-crossing spanning tree, and that it is hard to approximate the minimum number of crossings in a spanning tree. We give improved fixed-parameter algorithms, where we use the number k of crossing edge pairs and the number μ of crossing edges in the given graph as parameters. The running times of our algorithms are $O^*(k^{O(\sqrt{k})})$ and $O^*(\mu^{O(\mu^{2/3})})$, where the O^*-notation neglects polynomial terms. Our method can be applied to several other non-crossing subgraph problems in topological graphs.

1 Introduction

A *topological graph* is a graph with an embedding of its edges as curves in the plane [5] and each pair of edges intersect at most once. A topological graph is said to be *non-crossing* if none of the edge curves cross. We consider some problems that involve finding a non-crossing subgraph satisfying some property: spanning tree, s–t path, and cycle. All of these problems are known to be NP-hard. In this article we focus on the non-crossing spanning tree problem.

Let G be a topological graph on n vertices. A *crossing* is a pair of edges that meet in a non-vertex point, and a *crossing edge* is one that participates in some crossing. A *crossing point* is a non-vertex point that is contained in at least two edge curves. Note that if d edges intersect in a crossing point, they create $\binom{d}{2} = d(d-1)/2$ crossings. Given a topological graph G, let X be the set of crossings in G. We denote by E_X the set of crossing edges. Let $k = |X|$ be the number of crossings and let $\mu = |E_X|$ be the number of crossing edges. Observe that

[1]This work was started at the workshop *New Horizons in Computing* organized by Takao Asano in Chofu, Japan, February 2006.

[2]Partially supported by the project *New Horizons in Computing*, Grant-in-Aid for Scientific Research on Priority Areas, MEXT Japan.

[3]Supported by grant WO 758/4-2 of the German Research Foundation (DFG).

$\mu \leq 2k \leq \mu(\mu-1)$. A topological graph is embedded in the plane, with vertices located at particular points and edges corresponding to a topological path connecting the two endpoints. We refer to the embeddings of the vertices also as vertices, and to the topological paths as *curves*. We assume without loss of generality that the curves intersect only in individual points, not in curve segments.

Recently, Knauer et al. [2, 3] gave fixed-parameter algorithms in terms of the parameter k. They gave algorithms for all of these problems that run in $O^*(2^k)$ time, and improved this for the non-crossing spanning tree problem to $O^*(1.9999992^k)$ time and for non-crossing s–t paths and cycles to $O^*((\sqrt{3})^k)$ time, where the O^*-notation hides polynomial terms. They also showed that these subgraph problems are hard to approximate. Precisely speaking, they showed that for any $\epsilon > 0$ it is NP-hard to obtain a $k^{1-\epsilon}$-approximation, both for the number of crossings in a topological spanning tree and for the number of components in a non-crossing spanning forest.

In this paper we give algorithms with an improved time complexity of $k^{O(\sqrt{k})}$, which grows much more slowly than the $const^k$-terms in the running times of the algorithms known so far. We also consider the parameter μ, the number of crossing edges, and give an $\mu^{O(\mu^{2/3})}$-time algorithm. Our technique may have potential for solving various connectivity problems on planar (and nearly-planar) graphs; this is currently under investigation. We first massage the instance into a planar graph with desirable properties. We then apply the planar separator theorem [4] to divide the instance into subproblems. Finally we search for appropriate configurations of the subproblems to model the connectivity constraints.

Due to space limitation, we mainly focus on the non-crossing spanning tree problem (NCST) in this extended abstract, although we can treat the other problems analogously. In the next section we show how an instance of the NCST problem can be reduced to a more amenable one. We first show how to obtain a *kernel* of size $O(k)$, and then how we may assume we are dealing with an instance where only two edge cross in the same point and at most three edges meet in any vertex. Next we detail our algorithm based on finding small separators. Finally we indicate how the algorithm can also be used to give improved bounds in terms of the parameter μ and to solve optimization problems.

2 Reductions

We can obtain a small problem *kernel*, capturing all the intricacies of the problem, by expanding on an observation of Knauer et al. [3]. Recall that E_X is the set of crossing edges, and let V_X denote the set of its endpoints. Consider the connected components $C_1, C_2, \ldots, C_t$ of the graph $G_0 = (V, E \setminus E_X)$ obtained by removing the crossing edges.

We construct a graph $\hat{G}$ on V_X as follows. For each component C_i of G_0, which is non-crossing, we compute a spanning tree T_i, choose an arbitrary root, and traverse T_i in depth-first order. This yields an order of the vertices in $V_X \cap C_i$. We connect each (except the last) vertex to its successor by an edge. The embedding of each such edge $\{u, v\}$ follows the chain of curves on the unique path connecting u and v in T_i. Since C_i is non-crossing, the new edges do not cross (but they may overlap). Finally we add the edges in E_X to $\hat{G}$. The number of vertices

of $\hat{G}$ is at most 2μ, and the number of edges is at most μ plus the number of vertices (minus one). It is routine to show the following lemma.

Lemma 1 *The graph $\hat{G}$ contains a spanning forest with at most τ crossings and κ components if and only if G does.*

For the NCST problem, we apply the above lemma for $\kappa = 1$ and $\tau = 0$. Thus, with linear-time preprocessing, we may assume that the graph is of size linear in μ. In other words, we have the following theorem:

Theorem 1 *There is a kernel for NCST with at most 3μ edges and vertices.*

The *multiplicity* of a crossing is the number of pairs of edges that meet in the same point. Large multiplicity can confuse good algorithms, especially those based on separators, and the same can be said of high-degree vertices. Fortunately, as we will see, we can assume without loss of generality that crossings are of unit multiplicity and vertices of maximum degree 3.

Theorem 2 *Suppose there is an algorithm that solves NCST on degree-3 graphs with unit crossing multiplicity in time $T(k, \mu, n)$. Then, there is an algorithm for NCST for general topological graphs running in time $O(T(k, \mu, n))$. The same holds for optimization versions of the NCST problem.*

The basic idea is to clip edges at high-degree vertices and to replace the clipped stars by binary trees and to wiggle edge curves in order to avoid degenerate crossings.

3 Separator-Based Algorithm

We describe here our algorithm for the non-crossing spanning tree problem.

Finding a separator. Let H be a topological graph with k crossings and μ crossing edges. Our kernelization ensures that H contains at most 2μ edges and vertices in total. We can regard the drawing of H as the drawing of the planar graph that we get if we regard each crossing point as a new vertex. In other words, we form the plane graph $P = P_H$ containing the vertices of H as well as a vertex for each crossing, where the edges of P are the segments of the curves defining edges of H between curve endpoints and/or crossings. The number $n(P)$ of vertices of P is $2\mu + k$. The maximum vertex degree is four, the degree of the new crossing vertices. The number of edges of P is exactly $2k$ more than that of H (changing a crossing point to a vertex splits each of the two edges crossing the point).

An α-separator of a graph on n vertices is a subgraph whose removal splits the graph into components, each of cardinality at most αn. The planar separator theorem [4] says that one can find in linear time a set of $\beta\sqrt{n}$ vertices that is a 2/3-separator, where $\beta \leq 2\sqrt{2}$. The currently best separator bound has $\beta \leq 1.97$ due to Djidjev and Venkatesan [1]. We refer to a 2/3-separator simply as a separator.

In the planar graph P, each vertex corresponds to a point touched or crossed by at most three edges of H. Thus, a vertex separator of cardinality q in P corresponds to an edge separator of size at most $3q$ in H.

To match the definition of the vertices of P, we define the *crossing measure* $\nu(H)$ of H to be the number of vertices of P_H. We can see that ν is the number of incidences of crossing points and edges going through them in the original graph.

By removing the given edge set from H, we may accomplish much more in reducing the number of vertices of P_H, but at least the vertices of the separator of P_H are removed. Thus, after the removal of the edge set from H, each of the components will have a crossing measure of at most $2\nu/3$.

Theorem 3 *Let H be a topological graph with crossing measure ν. Then, there exists an edge set S of at most $\sigma(\nu) = 3\beta\sqrt{\nu}$ edges, defined by a topological curve T_S, such that the components resulting from the removal of S from H have crossing measure at most $2\nu/3$.*

Corollary 1 *There exists an edge set S of size $\sigma(\nu)$ edges to separate H into (not necessarily connected) subgraphs H_1 and H_2 such that $\max(\nu(H_1), \nu(H_2)) \leq 2\nu/3$.*

The Algorithm. The idea of the algorithms is as follows. We first find a small edge separator S to separate H into two subgraphs H_1 and H_2 as in Corollary 1. We then *guess* which of the edges of S will participate in the solution, i.e., in a non-crossing spanning tree. Let $S' \subseteq S$ denote this set (which is non-crossing by definition). For each edge in S', we remove all edges crossing it from the graph. Now, we consider the NCST problem in $H_1 \cup H_2 \cup S'$, where we use the convention that we have already removed from H_1 and H_2 all edges that cross edges in S'. Next we guess the *configuration* of S' within a spanning tree T of H; such a configuration contains subtrees connecting some of the endpoints of S' within H_i as well as the edges of S'.

In order to describe configurations, we consider colorings of S' by a set $C = \{c_1, c_2, \ldots c_{s'}\}$ of $s' = |S'|$ colors. We regard a coloring ψ as an equivalence class of maps from S' to C, where two maps are equivalent if one can be transformed into the other by a permutation of colors. Consider a pair ψ, ψ' of colorings of S', and define a graph N whose vertices are the edges in S' and whose edges are pairs (e, e') of edges of S' with $\psi(e) = \psi(e')$ or $\psi'(e) = \psi'(e')$. We say that ψ and ψ' are *complementary* if N forms a connected graph. Given a spanning forest F_1 (resp. F_2) of H_1 (resp. H_2), we define a coloring ϕ_{F_1} (resp. ϕ_{F_2}) such that two edges of S' have the same color if and only if their endpoints in H_1 (resp. H_2) are in a same connected component in F_1 (resp. F_2). It is straightforward to show the following lemma.

Lemma 2 *For a pair of spanning forests F_1 of H_1 and F_2 of H_2, the union $F_1 \cup F_2 \cup S'$ contains a spanning tree of H if and only if ϕ_{F_1} and ϕ_{F_2} are complementary.*

Now, our task is to find non-crossing spanning forests F_1 of H_1 and F_2 of H_2 such that ϕ_{F_1} and ϕ_{F_2} are complementary. Given a coloring ψ, we define the graph $H_1 \odot \psi$ by adding to H_1 the edges in S' and joining edges of S' colored by c_i to an artificial vertex v_i for each color c_i. We define $H_2 \odot \psi$ analogously. Again, proving the following lemma is routine.

Lemma 3 *If $H_1 \odot \psi$ has a non-crossing spanning tree T, the forest F obtained by restricting T to H_1 satisfies that ψ and ϕ_F are complementary. Conversely, if there is a non-crossing forest F of H_1 such that ψ and ϕ_F are complementary, $H_1 \odot \psi$ has a non-crossing spanning tree.*

Our algorithm is as follows. For each coloring ψ, we seek a non-crossing spanning tree in $H_1 \odot \psi$ (this is done recursively). If there is a non-crossing spanning tree T in $H_1 \odot \psi$, we consider the forest F_1 obtained by restricting T to H_1, and compute $\psi' = \phi_{F_1}$. Then, we seek a non-crossing spanning tree in $H_2 \odot \psi'$. If we find a tree T', we output a spanning tree of the union of T' and F_1. Since both T' and F_1 are non-crossing, the output tree is non-crossing.

If the algorithm terminates and outputs no tree, we decide that the original graph does not contain a non-crossing spanning tree.

Correctness. It is clear that if the algorithm outputs a tree, it is a non-crossing spanning tree of H. Thus, it remains to show that if the algorithm declares that there is no non-crossing spanning tree, H indeed has no non-crossing spanning tree.

Suppose that H has a non-crossing spanning tree T, and let F_1 and F_2 be forests obtained by restricting T to H_1 and H_2, respectively. We first consider $\psi = \phi_{F_2}$, and find a non-crossing spanning tree T_1' in $H_1 \odot \psi$. We can always find one, since we can construct such a tree by contracting each connected component of F_2 to an artificial vertex in $H_1 \odot \psi$. Let F_1' be the restriction of T_1' to H_1. Note that F_1' may have a different topology from F_1. Then, we consider $\psi' = \phi_{F_1'}$ in the algorithm. We claim that $H_2 \odot \psi'$ always has a non-crossing spanning tree.

By Lemmas 2 and 3, ψ and ψ' are complementary since T_1' is a non-crossing spanning tree of $H_1 \odot \psi$. Thus, again by Lemma 3, $H_2 \odot \psi$ has a non-crossing spanning tree. Thus, the claim is proven.

Complexity. Recall that the size $s = |S|$ of the edge separator S is at most $\sigma(n(P))$. We check all 2^s subsets of S. For each of the at most 2^s non-crossing candidates for S', we consider its colorings. For each candidate the number of colorings is bounded by the Bell number, which is at most s^s. Hence, we derive at most $(2s)^s$ subproblems, and solve each of them recursively. Note that for each coloring ψ of S', we solve (at most) one problem on each of H_1 and H_2. To be precise, H_1 and H_2 are slightly modified by artificial nodes and by S', but we can ignore the effect of this modification in the analysis.

Recall the crossing measure $\nu(G) = 2\mu(G) + k(G)$, the number of nodes in the graph P. Each subproblem has a measure of at most $2\nu(G)/3$. Let $T(\nu)$ be the number of subproblems generated for an instance of measure ν. In the worst case, we obtain two subproblems, one of measure $2\nu/3$ and the other of measure $\nu/3$. Hence,

$$T(\nu) \leq (2s)^s(T(2\nu/3) + T(\nu/3)) + 1 \leq 2(2s)^s T(2\nu/3),$$

where $s = \sigma(\nu) \leq 6\sqrt{\nu}$. Then, this is satisfied by $T(\nu) = 2(2\sigma(\nu))^{q\sigma(\nu)}$, where $q = 1 + \sqrt{2/3}$. It follows that the time complexity of the algorithm is $O^*(2^{O(\sqrt{k}\log k)}) = O^*(k^{O(\sqrt{k})})$.

4 The parameter μ

We can design a straightforward algorithm of time complexity $O^*(2^\mu)$ for the NCST problem, since we can guess the set of edges in the spanning tree among all crossing edges. The $O^*(2^{\sqrt{k}\log k})$ bound is worse if $k > \mu^2/\log^2 \mu$. However, we can obtain a non-trivial bound in terms of the parameter μ. To achieve this, we split the computation into two cases, depending on the size of μ relative to k. Let $R(\mu)$ be the number of subproblems in an instance with μ crossing edges.

Case 1: $k < 2\mu^{4/3}$, i.e., μ is large. Then the separator-based algorithm gives $R(\mu) < ck^{\sqrt{k}} < c\mu^{2\mu^{2/3}}$. Thus, we can use induction. We ignore c and set it to be 1 for simplicity, and show that $R(\mu) < F(\mu) = \mu^{\mu^{2/3}}$.

Case 1: $k \geq 2\mu^{4/3}$. Then there exists an edge that participates in at least $\sqrt{\mu} \geq k^{1/3}$ crossings. We do a bounded search on that edge, resulting in two subproblems: one without that edge, and the other without all the edges crossing it. This gives the recurrence

$$R(\mu) \leq R(\mu - 1) + R(\mu - 2\mu^{1/3}) + 1.$$

Then,

$$R(\mu-1) < (\mu-1)^{(\mu-1)^{2/3}} < (\mu-1)[\mu^{(\mu-1)^{2/3}-1}] < (\mu-1)[\mu^{\mu^{2/3}-1}].$$

Thus, $F(\mu) - R(\mu-1) > \mu^{\mu^{2/3}-1}$. On the other hand, $R(\mu - 2\mu^{1/3}) < \mu^{(\mu-2\mu^{1/3})^{2/3}}$. Since we can assume that μ is large, say, larger than 10, we have

$$(\mu - 2\mu^{1/3})^{2/3} = \mu^{2/3}(1 - 2\mu^{-2/3})^{2/3} = \mu^{2/3}[1 - 4\mu^{-2/3}/3 + o(\mu^{-1})].$$

Now,

$$F(\mu) - R(\mu-1) - R(\mu - 2\mu^{1/3}) > \mu^{\mu^{2/3}-1} - \mu^{\mu^{2/3}-4/3+o(1)} > 1,$$

assuming μ is large. Thus, we have $F(\mu) > R(\mu-1) + R(\mu - 2\mu^{1/3}) + 1 \ \geq \ R(\mu)$.

Theorem 4 *The problem NCST can be solved in time $\mu^{O(\mu^{2/3})}$.*

Concluding Remarks

As we have claimed in the introduction, we can apply our method to several other problems such as non-crossing s–t paths and cycles. We can also deal with the optimization problems, minimizing either the number of components in a non-crossing spanning forest or the number of crossing edges in a spanning tree. These extensions will be given in the full paper.

References

[1] H. N. Djidjev and S. M. Venkatesan. Reduced constants for simple cycle graph separation. *Acta Informatica*, 34:231–243, 1997.

[2] C. Knauer, É. Schramm, A. Spillner, and A. Wolff. Configurations with few crossings in topological graphs. In X. Deng and D.-Z. Du, editors, *Proc. 16th Ann. Int. Symp. Algorithms Comput. (ISAAC'05)*, vol. 3827 of *LNCS*, pp. 604–613. Springer-Verlag, 2005.

[3] C. Knauer, É. Schramm, A. Spillner, and A. Wolff. Configurations with few crossings in topological graphs. *Computational Geometry: Theory and Applications*, 2006. To appear.

[4] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math.*, 36:177–189, 1979.

[5] J. Pach and G. Tóth. Unavoidable Configurations in Complete Topological Graphs. *Proc. Graph Drawing 2000.*, LNCS 1984 :328–337, 2001.