Spanning trees across axis-parallel segments

Michael Hoffmann* Csaba D. Tóth†

Abstract. Given a set P of points and a set S of pairwise disjoint axis-parallel line segments in the plane, we construct a straight line spanning tree T on P such that every segment in S crosses at most three edges of T.

1 Introduction

Geometric shortest paths and shortest spanning trees are among the most well studied topics in computational geometry [7]. In particular, shortest paths have been studied in the presence of obstacles, where a path has to avoid all obstacles [3, 9]. A relaxed concept of obstacles is considered in the weighted shortest path problem [2, 4, 8], where in principle any region can be entered, but a path is penalized for traversing heavy regions.

We study a problem introduced by Asano et al. [1]: given a set P of points and a set S of barriers in the plane, what is the minimum cost of a *straight line* spanning tree, where the cost of an edge is the number of barriers crossed, and the cost of a graph is total cost of its edges.

Asano et al. [1] reduced several variants of the problem to the case that the barriers are disjoint line segments in the plane. They studied the quantity $\mathrm{maxcr}(S,T)$, which is the maximum number of edges of a graph T that intersects any barrier in S. They showed that there is a spanning tree with $\mathrm{maxcr}(S,T) \leq 4$; it follows that the minimum cost of a spanning tree is at most $4|S|$. The worst case lower bound for the total cost is $2|S|$. On the other hand, there are points and disjoint segments such that $\mathrm{maxcr}(S,T) = 3$ [5]. This lower bound construction can also be realized with axis-parallel line segments. Recently, Krumme et al. [6] proved that if the segment barriers form a convex subdivision and there is a point in each convex cell, then there is a spanning tree with $\mathrm{maxcr}(S,T) \leq 2$, which is best possible. We prove a worst-case optimal bound on $\mathrm{maxcr}(S,T)$ in the case that the barriers have only two distinct orientations.

Theorem 1 *Given a set S of disjoint axis-parallel line segments and a point set P in the plane, one can construct a spanning tree T over P with* $\mathrm{maxcr}(S,T) \leq 3$.

Our proof is constructive. However, we leave it as an open problem to devise an optimal algorithm for constructing a *minimum* cost spanning tree.

*Institute for Theoretical Computer Science, ETH Zürich, `hoffmann@inf.ethz.ch`

†Department of Mathematics, MIT, `toth@math.mit.edu`

2 Construction of a spanning tree

This approach is similar to that of Asano et al. [1]. In fact, we first present a simplified analysis of an algorithm in [1]; and then extend it (with a coloring scheme) to prove Theorem 1. Assume for simplicity that the input (P,S) is in *general position*, that is, there are no three collinear points among the points of P and the segment endpoints of S.

Denote the points of P by $p_0, p_1, \ldots, p_{n-1}$ ($n = |P|$) in order of non-decreasing y-coordinate. Let p_{-2} and p_{-1} be the lower left and the lower right corner of an axis-parallel box that contains P in its interior. We start with an initial tree T_0 on the vertex set $V(T_0) = \{p_{-2}, p_{-1}, p_0\}$ with edge set $E(T_0) = \{p_{-2}p_0, p_{-1}p_0\}$. Our algorithm proceeds in $n-1$ phases: in phase i, $1 \leq i \leq n-1$, we have a spanning tree T_i on $\{p_{-2}, p_{-1}, \ldots, p_{i-1}\}$. Points p_{-2} and p_{-1} are artificial vertices; if both are leaves of a tree T, however, then they can be removed without increasing $\mathrm{maxcr}(S,T)$.

Let π_i denote the unique path in the tree T_i between p_{-2} and p_{-1}. Since $p_{-2}p_{-1}$ is a side of the bounding box, the curve $\pi_0 \cup p_{-2}p_{-1}$ encloses a simply connected region, which we call the *core* of T_i and denote it by C_i. We maintain the following invariants for the spanning tree T_i at every phase i, $0 \leq i \leq n-1$.

(α) $V(T_i) = \{p_{-2}, p_{-1}, p_0, \ldots, p_i\}$;

(β) p_{-2} and p_{-1} are leaves of T_i;

(γ) the planar drawing of T_i lies in C_i;

(δ) $C_{i-1} \subset C_i$, for every $i = 1, 2, \ldots, n-1$.

2.1 Visibility

We define visibility with respect to a straight line graph G and a set S of segments in the plane. The edges of G and the segments that cross any edge of G are opaque, all other segments are transparent. Specifically, let $\hat{S}_G$ denote the set of opaque segments which cross some edge of G. We show that if a point p is separated from G by a line, then p sees a point on an edge of G.

Lemma 2 *Consider a geometric graph G with at least one edge, a set of segments S, and a point p in the plane. Assume that the vertical line through p intersects G, but the horizontal line though p is strictly above G. Then there is a point q in the relative interior of an edge $e \in E(G)$ such that p sees q.*

Proof. We show that Algorithm 1 below finds a point visible from p on one of the edges of G.

Algorithm 1 `visibility`(G, S, p)

1. *Let w_0 be a point vertically below p that p sees. Let $j = 0$.*
2. *Repeat until w_j lies on an edge of G:*
   (a) *Let t_j be a common point of G and the segment containing w_j, and let $w_j w_{j+1}$ be the longest segment contained in $w_j t_j$ such that p sees every point in* $\mathrm{relint}(w_j w_{j+1})$.
   (b) *If p does not see w_j, then let w_{j+1} be the segment endpoint that occludes w_j from p.*
   (c) *Put $j := j + 1$*

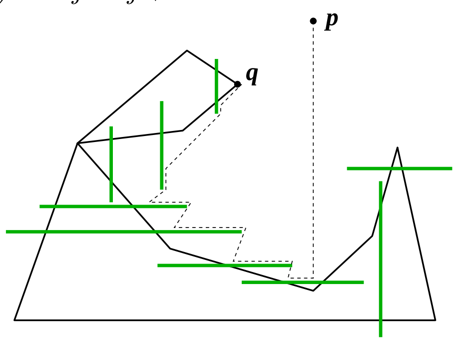

Figure 1: Path ω from p to q in Algorithm 1.

The path $\omega = (w_0, w_1, w_2, \ldots)$ is composed of portions of opaque segments visible from p and portions of rays emanating from p (Fig. 1). This implies that ω winds either entirely clockwise or counterclockwise around p. If ω is a cycle, then it circumscribes p. Let w_a be the point of ω with maximal y-coordinate. On one hand, segment $w_{a-1}w_a$ cannot be a portion of a segment $s \in \hat{S}_G$ (step 2a), because it is not approaching any intersection point $t_i \in G$ that lies below p. On the other hand, $w_{a-1}w_a$ cannot lie along pw_{a-1} (step 2b), otherwise w_{a-1} would be above w_a. We conclude that ω is not a cycle, its last vertex is visible from p and lies on an edge of G. Finally, notice that if a vertex $v \in V(G)$ is visible from p, then a point in the relative interior of an edge incident to v is also visible to p due to the general position assumption. $\square$

2.2 Main Algorithm

In phase i of our algorithm, visibility is limited by the tree T_{i-1} and $\hat{S}_i$, where $\hat{S}_i$ denotes the set of segments that cross some edge of T_{i-1}. The analysis of our algorithm focuses on the portions of the segments $\hat{S}_i$ not contained in the core C_i. A *spike* for T_i is a maximal continuous portion $\hat{s}$ of a segment $s \in \hat{S}_i$ that lies in the exterior of the core C_i. A *mid-spike* has two endpoints on π_i, an *end-spike* has one endpoint on π_i (and the other endpoint is an endpoint of a segment s).

The following algorithm constructs a spanning tree $T = T_{n-1}$ on P by successively attaching new vertices to T_i, $i = 0, 1, \ldots, n-1$.

Algorithm 2 *Input: point set P and segment set S. Initiate T_0 by letting $V(T_0) = \{p_{-2}, p_{-1}, p_0\}$ and $E(T_0) = \{p_{-2}p_0, p_{-1}p_0\}$.*
For $i = 1$ to $n - 1$ do

1. *Choose an edge $u_i v_i \in E(T_{i-1})$ such that p_i sees a point $q_i \in \mathrm{relint}(u_i v_i)$.*
2. *Find two vertices $u'_i \in \pi_{i-1}$ and $v'_i \in \pi_{i-1}$ (the choice of u'_i and v'_i is described below).*
3. *Put $E(T_i) = E(T_{i-1}) \cup \{p_i u'_i, p_i v'_i\} \setminus \{u_i v_i\}$ and $V(T_i) = V(T_{i-1}) \cup \{p_i\}$.*

Removing the edge $u_i v_i$ disconnects the tree T_{i-1} (and the path π_{i-1}). Hence, the points u'_i and v'_i have to be in different components. Note, also, that $\mathrm{relint}(p_i u'_i)$ and $\mathrm{relint}(p_i v'_i)$ have to be disjoint from T_{i-1}. We describe the choice of u'_i and v'_i in Section 3.

2.3 Arbitrarily oriented segments

Algorithm 2 is used by Asano et al. [1] in the case of arbitrarily oriented segments. Their choice of u'_i and v'_i can be summarized as follows. For two points p and q, not in the interior of the core C_i, let $g(p, q)$ denote the shortest path between p and q that is disjoint from $\mathrm{int}(C_i)$. Consider the pseudo-triangle P_i formed by $u_i v_i$, $g(p_i, u_i)$, and $g(p_i, v_i)$. Let u'_i and v'_i be the vertices of T_{i-1} adjacent to p_i in $g(p_i, u_i)$ and $g(p_i, v_i)$, respectively.

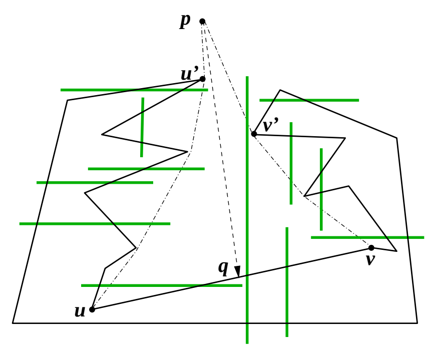

Figure 2: Pseudo-triangle P formed by uv and the geodesics $g(p, u)$ and $g(p, v)$.

Lemma 3 *With this choice of u'_i and v'_i, every $s \in S$ crosses at most 4 edges of every tree T_i, $i = 0, 1, ..., n-1$.*

Proof. Consider a segment $s \in S$. If it is first crossed in phase i, then it is crossed at most twice (by $p_i u'_i$ or $p_i v'_i$), and it has one or two spikes in the exterior of C_i. Now, look at a spike $\hat{s} \subset s$ in a subsequent phase $j > i$. Since $p_j q_j$ does not cross any spikes, every end-spike $\hat{s}$ crosses only one of $p_j u'_j$ and $p_j v'_j$. No mid-spike gets any new crossings. Since P_j is a pseudo-triangle, if $\hat{s}$ crosses $p_j u'_j$ or $p_j v'_j$, then its endpoint must lie in the interior of C_j, hence $\hat{s} \setminus C_j$ becomes a mid-spike. In summary, every end-spike can be crossed at most once more and no mid-spike is crossed again. This implies that every segment in S crosses at most 4 edges, and $\mathrm{maxcr}(S, T_i) \leq 4$ throughout Algorithm 2. $\square$

3 Axis-parallel segments

We extend the above argument by color-coding the spikes. When a segment $s \in S$ is first crossed in phase i of Algorithm 2, we color its spikes red, blue, and green so that (1) at most one of its spikes is blue and (2) a spike is green only if it it the unique spike of s. We then assure that no red spike gets new crossings, every blue spike gets at most one new crossing, and every green spike gets at most two new crossings. This immediately implies that every segment in S crosses at most 3 edges of T_i for $i = 0, 1, \ldots, n-1$.

For the definition of the color scheme, we need to distinguish upper and lower edges along the path π_i. An edge $e \subset \pi_i$ is an *upper* (*lower*) edge if the core C_i lies below (above) e.

3.1 Color scheme for new spikes

In phase i of the algorithm, every spike has some color. We assign a color to a spike $\hat{s}$, $\hat{s} \subset s$, at the phase i where s first intersects an edge of T_i. In later phases (even though s may cross new edges of T_j, $j > i$, and its spikes may be shortened), every spike retains its color.

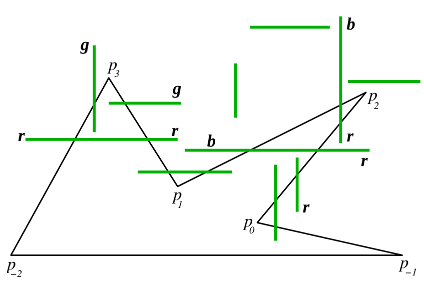

Figure 3: An example for the coloring scheme.

Consider an end-spike $\hat{s}$ of an edge $s \in S$ such that s crosses an edge of T_i but it is disjoint from C_{i-1}. Color $\hat{s}$ *red* if it is incident to a *lower* edge. Color $\hat{s}$ *green* if $\hat{s}$ is the only spike of s and it is incident to an upper edge. Now assume that s crosses both $p_i u_i'$ and $p_i v_i'$, and so s has two spikes. Color $\hat{s}$ *blue* if the other spike along s is incident to a lower edge; otherwise color it *red*. See Fig. 3for an example. Note that every upper spike of a vertical segment is either green or blue.

3.2 Finding vertices u_i' and v_i'

We start with the same vertices u_i' and v_i' as in Subsection 2.3, then we move them until the edges $p_i u_i'$ and $p_i v_i'$ do not cross any red spikes. We describe this subroutine for u' only, it is analogous for v'. (See Fig. 4.)

Algorithm 3

1. *Initially, put $j = 0$ and let h_0 be the neighbor of p_i in $g(p_i u_i)$.*
2. *Repeat until every red or mid-spike that crosses $p_i h_j$ crosses $u_i v_i$, too:*
   (a) *Let z_1 be an intersection of $p_i h_j$ with a red or mid-spike (which does not intersect $u_i v_i$) closest to p_i.*
   (b) *Let z_2 be the endpoint of this spike on π_{i-1} such that $\angle z_1 p_i q_i \subset \angle z_2 p_i q_i$.*
   (c) *Let z_3 be the vertex of the edge of π_{i-1} that contains z_2 such that z_3 and p_i are on the same side of the line through the spike.*
   (d) *Let h_{j+1} be the vertex adjacent to p_i along the shortest path $g(p_i, z_3)$, and put $j := j + 1$.*
3. *Output $u_i' := h_j$.*

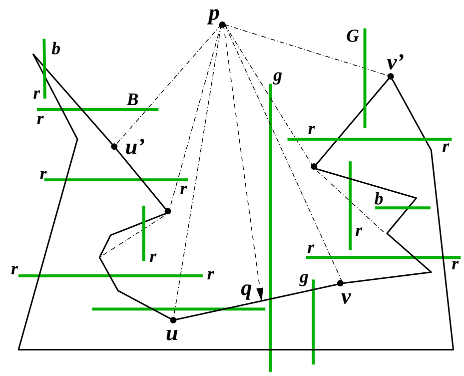

Figure 4: Algorithm 3 moves from u and v to u' and v'.

Algorithm 3 terminates because in every loop, the angle $\angle q p_i h_j$ strictly increases. (Notice that the angle in step 2b increases indeed always because no upper end-spike is red.)

Proposition 4 *If $p_i h_j$ crosses a red or mid-spike $\hat{s}$ in Algorithm 3, then $p_i h_{j+1}$ does not cross $\hat{s}$.*

Proof. The polygonal curve (p_1, z_1, z_2, z_3) is either convex or reflex. Thus, $g(p_i, z_3)$ is disjoint from $z_1 z_2$. □

Proposition 4 implies that red and mid-spikes do not get new crossings in Algorithm 2. It remains to show that green and blue spikes do not collect too many new crossings.

Proposition 5 *In Algorithm 3, the y-coordinate of every h_{j+1}, $j \geq 0$, is bigger than that of h_j.*

Proof. Consider iteration j of Algorithm 3. Segment $z_1 z_2$ lies along a red or mid-spike (in particular, it is not part of any upper end-spike). Hence z_1 lies below p_i (recall that p_i has greater y-coordinate than any point of T_{i-1}). Point h_j lies below z_1 since $z_1 \in p_i h_j$. It is enough to show that h_{j+1} lies above z_1.

The geodesic curve $g(p_i, z_3)$ and h_{j+1} lies in the angular domain $\angle p_i z_1 z_2$. Recall that $z_1 z_2$ lies along a segment of S, and so it is axis-parallel. If $z_1 z_2$ is horizontal, then the angular domain $\angle p_i z_1 z_2$ is above $z_1 z_2$. If $z_1 z_2$ is vertical, then it lies along a lower or a mid-spike, and z_2 lies above z_1. Hence, the angular domain $\angle p_i z_1 z_2$ is also above z_1. □

3.3 The position of the edge visible to p_i

We note an important property of the point q that p sees in Algorithm 1 in case all segments are axis-parallel.

Proposition 6 *If S is a set of disjoint axis-parallel segments, then one of the following holds for uv and $q \in \mathrm{relint}(uv)$ claimed by Lemma 2 (see Fig. 5):*
(1) p and q have the same x-coordinates;
(2) q has larger (smaller) x-coordinate than p and uv is an upper edge with positive (negative) slope;
(3) q has smaller (larger) x-coordinate than p and uv is a lower edge with positive (negative) slope.

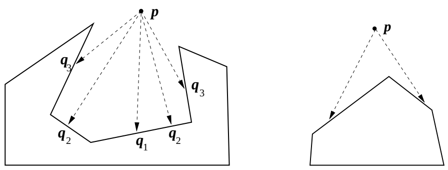

Figure 5: Relative positions of point q, edge e, and the core (left); and positions that do not occur (right).

3.4 Backhand and forehand crossings

Recall that every segment $s \in S$ has at most one green or blue spike. The green or blue spike $\hat{s}$, $\hat{s} \subset s$, can possibly be crossed by $p_i u_i'$ or $p_i v_i'$ (which is a *new* crossing only if $\hat{s}$ is not incident to $u_i v_i$). Ideally, the common endpoint of s and $\hat{s}$ lies in $\mathrm{int}(C_i \setminus C_{i-1})$. We call such a new crossing a *forehand crossing* (e.g., spike B in Fig. 4). In this case, the remaining spike $\hat{s} \setminus C_i$ is a mid-spike, which cannot get any new crossings. If the common endpoint of $\hat{s}$ and s remains in the exterior of C_i, we talk about a *backhand crossing* (e.g., spike G in Fig. 4).

Proposition 7 *Assume that in step i of Algorithm 2, a blue or green spike $\hat{s}$ is incident to an edge $a_{i-1}b_{i-1} \in \pi_{i-1}$, and $p_i u_i'$ (or $p_i v_i'$) crosses $\hat{s}$ backhandedly. Then there is a step $j \in \mathbb{N}$ in Algorithm 3 where $p_i h_j$ crosses a horizontal red spike which is incident to $a_{i-1}b_{i-1}$.*

Proof. Recall that $p_i q_i$ does not cross any spikes, and $p_i u_i$ does not cross backhandedly any spike (c.f. Lemma 3). Similarly in Algorithm 3, if $p_i h_{j-1}$ does not cross spike $\hat{s}$ backhandedly but $p_i h_j$ does, then $\hat{s}$ must be incident to $z_2 z_3 \subset \pi_{i-1}$. $\square$

Proposition 8 *No horizontal spike is crossed backhandedly.*

Proof. Assume that $p_i u_i'$ or $p_i v_i'$ crosses a blue or green horizontal spike $\hat{s}$ backhandedly. Let $a_{i-1}b_{i-1} \in \pi_{i-1}$ denote the edge incident to $\hat{s}$ such that b_{i-1} has larger y-coordinate. By Proposition 7, $\hat{s}$ can be crossed backhandedly only if a red spike $\hat{t}$ incident to $a_{i-1}b_{i-1}$ intersects $p_i h_j$ for some $j \in \mathbb{N}$. Since h_{j+1} is on the shortest path $g(p_i, b_{i-1})$, $p_i h_{j+1}$ cannot cross $\hat{s}$. By Proposition 5, $p_i u_i'$ does not cross $\hat{s}$, either. A contradiction. $\square$

Lemma 9 *Every vertical segment in $s \in S$ crosses at most three edges of T_i, $i = 0, 1, \ldots, n-1$.*

Proof. *Case 1: Initially, s has two spikes.* The lower (red) spike cannot get any new crossing by Proposition 4. We show that the upper (blue) spike $\hat{s}$ gets at most one new crossing. Since any forehand crossing turns $\hat{s}$ into a mid-spike, it is enough to show that it cannot be crossed backhandedly. Assume, to the contrary, that $p_i u_i'$ or $p_i v_i'$ crosses $\hat{s}$ backhandedly. By Proposition 7, the (upper) edge $a_{i-1}b_{i-1} \in \pi_{i-1}$ incident to $\hat{s}$ is also incident to a red horizontal spike, which intersects some $p_i h_j$, $j \in \mathbb{N}$.

Suppose that spike $\hat{s}$ was created in a phase k, $0 \leq k < i$, and it was incident to the edge $p_k u_k'$. We first argue that $p_k u_k'$ was not incident to any red spike: Since s has a lower spike, an upper edge $p_k u_k'$ and a lower edge $u_k v_k'$ were added to T_{k-1}, and so every new spike incident $p_k u_k'$ is colored green or blue. If there was any horizontal red spike incident to $u_k v_k$, it cannot cross $p_k u_k'$ because the slopes of $u_k v_k$ and $p_k u_k'$ have different signs by Proposition 6 (c.f., Fig. 5).

Denote by $a_\ell b_\ell$ the edge of T_ℓ incident to $\hat{s}$ in a later phase ℓ, $k < \ell \leq i$. If a new red horizontal spike $\hat{t}$ appears on $a_\ell b_\ell$, then the intersection point $\hat{t} \cap a_\ell b_\ell$ lies above $\hat{s} \cap a_\ell b_\ell$, because the lower endpoint of s is below $a_\ell b_\ell$ and s and t are disjoint. So, either $\hat{t}$ lies above $\hat{s}$ and so an edge $p_i h_{j+1}$ is above $\hat{s}$; or $\hat{t}$ is entirely on the left (right) side of $\hat{s}$ and $p_i h_{j+1}$ also remains entirely on the same side of $\hat{s}$. In either case, $p_i h_{j+1}$ cannot cross $\hat{s}$: a contradiction.

Case 2: Initially, s has one spike only. It is enough to show that the upper spike $\hat{s}$ can be crossed backhandedly at most once. Assume that $\hat{s}$ is crossed backhandedly in phase k. We can repeat the argument of the previous paragraph: For any new red horizontal spike $\hat{t}$, the intersection $\hat{t} \cap a_\ell b_\ell$ lies above $\hat{s} \cap a_\ell b_\ell$, because the lower endpoint of s is below $a_\ell b_\ell$. $\square$

References

[1] T. Asano, M. de Berg, O. Cheong, L. J. Guibas, J. Snoeyink, & H. Tamaki, Spanning trees crossing few barriers, *Discrete Comput. Geom.* **30** (2003), 591–606.

[2] D.Z. Chen, K.S. Klenk, H. T. Tu, Shortest path queries among weighted obstacles in the rectilinear plane, *SIAM J. Comput.* **29** (4) (2000), 1223–1246.

[3] P.J. de Rezende, D.T. Lee, and Y.F. Wu, Rectilinear shortest paths in the presence of rectangular barriers, *Discrete Comput. Geom.* **4** (1989), 41–53.

[4] L. Gewali, A. Meng, J.S.B. Mitchell, S.C. Ntafos, Path planning in $0/1/\infty$ weighted regions with applications, in *Proc. 4th SoCG*, ACM Press, 1988, 266–278.

[5] M. Hoffmann and Cs. D. Tóth, Connecting points in the presence of obstacles in the plane, in: *Proc. 14th Canadian Conf. Comput. Geom. (Lethbridge, AB, 2002)*, 63–67.

[6] D. Krumme, E. Rafalin, D.L. Souvaine, and C.D. Tóth, Tight bounds for connecting sites across barrieres, in *Proc. 22nd SoCG*, ACM Press, 2006, 439–448.

[7] J.S.B. Mitchell, Shortest paths and networks, in *Handbook of Discrete and Computational Geometry*, CRC Press, Boca Raton, FL, 1997, 445–466.

[8] J.S.B. Mitchell and C.H. Papadimitriou, The weighted region problem: Finding shortest paths through a weighted planar subdivision, *J. ACM* **38** (1991), 18–73.

[9] Y.-F. Wu, P. Widmayer, M. D. F. Schlag, and C. K. Wong, Rectilinear shortest paths and minimum spanning trees in the presence of rectilinear obstacles, *IEEE Trans. Comput.*, **36** (1987), 321–331.