Maximum Plane Trees in Multipartite Geometric Graphs

Ahmad Biniaz[1] Prosenjit Bose[1] Kimberly Crosbie[1] Jean-Lou De Carufel[2]
David Eppstein[3] Anil Maheshwari[1] Michiel Smid[1]

July 7, 2017

Abstract

A geometric graph is a graph whose vertices are points in the plane and whose edges are straight-line segments between the points. A plane spanning tree in a geometric graph is a spanning tree that is non-crossing. Let R and B be two disjoint sets of points in the plane where the points of R are colored red and the points of B are colored blue, and let $n = |R \cup B|$. A bichromatic plane spanning tree is a plane spanning tree in the complete bipartite geometric graph with bipartition (R, B). In this paper we consider the maximum bichromatic plane spanning tree problem, which is the problem of computing a bichromatic plane spanning tree of maximum total edge length.

1. For the maximum bichromatic plane spanning tree problem, we present an approximation algorithm with ratio 1/4 that runs in $O(n \log n)$ time.
2. We also consider the multicolored version of this problem where the input points are colored with $k > 2$ colors. We present an approximation algorithm that computes a plane spanning tree in a complete k-partite geometric graph, and whose ratio is 1/6 if $k = 3$, and 1/8 if $k \geqslant 4$.
3. We also revisit the special case of the problem where $k = n$, i.e., the problem of computing a maximum plane spanning tree in a complete geometric graph. For this problem, we present an approximation algorithm with ratio 0.503; this is an extension of the algorithm presented by Dumitrescu and Tóth (2010) whose ratio is 0.502.
4. For points that are in convex position, the maximum bichromatic plane spanning tree problem can be solved in $O(n^3)$ time. We present an $O(n^5)$-time algorithm that solves this problem for the case where the red points lie on a line and the blue points lie on one side of the line.

1 Introduction

Let P be a set of n points in the plane in general position, i.e., no three points are collinear. Let $K(P)$ be the *complete geometric graph* with vertex set P. It is well known that the standard minimum spanning tree (MinST) problem in $K(P)$ can be solved in $\Theta(n \log n)$ time. Also, any minimum spanning tree in $K(P)$ is *plane*, i.e., its edges do not cross each other. The *maximum spanning tree* (MaxST) problem is the problem of computing a spanning tree in $K(P)$ whose total edge length is maximum. Monma *et al.* [5] showed that this problem can be

[1]School of Computer Science, Carleton University. Supported by NSERC.
ahmad.biniaz@gmail.com, kimberlycrosbie@cmail.carleton.ca, {jit, anil, michiel}@scs.carleton.ca
[2]School of Electrical Engineering and Computer Science, University of Ottawa. Supported by NSERC.
jdecaruf@uottawa.ca
[3]Computer Science Department, University of California, Irvine. Supported by NSF grant CCF-1228639.
eppstein@ics.uci.edu

solved in $\Theta(n \log n)$ time. However, a MaxST is not necessarily plane. Alon *et al.* [1] started the *maximum plane spanning tree* (MaxPST) problem, which is the problem of computing a plane spanning tree in $K(P)$ whose total edge length is maximum. It is not known whether or not this problem is NP-hard. They presented an approximation algorithm with ratio 0.5 for this problem. This approximation ratio was improved to 0.502 by Dumitrescu and Tóth [4].

Let R and B be two disjoint sets of points in the plane such that $R \cup B$ is in general position, and let $n = |R \cup B|$. Suppose that the points of R are colored red and the points of B are colored blue. Let $K(R, B)$ be the *complete bipartite geometric graph* with bipartition (R, B). The *minimum bichromatic spanning tree* (MinBST) problem is to compute a minimum spanning tree in $K(R, B)$. The *maximum bichromatic spanning tree* (MaxBST) problem is to compute a spanning tree in $K(R, B)$ whose total edge length is maximum. Recently, Biniaz *et al.* [2] showed that both the MinBST and the MaxBST problems can be solved in $\Theta(n \log n)$ time. We note that none of MinBST and MaxBST is necessarily plane; they might have crossing edges. Borgelt *et al.* [3] studied the problem of computing a minimum bichromatic plane spanning tree, which we refer to as the MinBPST problem. They showed that this problem is NP-hard, and also presented a polynomial-time approximation algorithm with approximation ratio of $O(\sqrt{n})$. In this paper we study the problem of computing a maximum bichromatic plane spanning tree, which we refer to as the MaxBPST problem. See Figure 1.

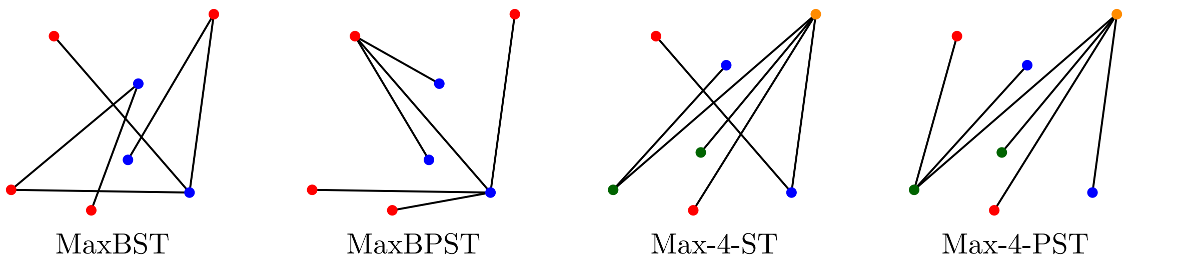


Figure 1: Maximum spanning trees.

A natural extension of the MinBST and the MaxBST problems is to have more than two colors. In this multicolored version, the input points are colored by $k > 2$ colors, and we are looking for a minimum (resp. maximum) spanning tree in which the two endpoints of every edge have distinct colors. In other words, we look for a minimum (resp. maximum) spanning tree in a complete k-partite geometric graph. We refer to these problems as the Min-k-ST and the Max-k-ST problems, respectively. Biniaz *et al.* [2] showed that both these problems can be solved in $O(n \log n \log k)$ time. Notice that the MinST and the MaxST problems are special cases of the Min-k-ST and the Max-k-ST problems in which every input point has a unique color, i.e., $k = n$. In this paper we also study the problem of computing a plane Max-k-ST, which we refer to as the Max-k-PST problem. See Figure 1.

1.1 Our contributions

In this paper we study the maximum plane spanning tree problems. In Section 3 we present an approximation algorithm with ratio 1/4 for the MaxBPST problem. We study the Max-k-PST problem in Section 4. For this problem, we present an approximation algorithm whose ratio is 1/6 if $k = 3$, and 1/8 if $k \geqslant 4$. In Section 5 we consider the MaxPST problem, where we modify the algorithm presented by Dumitrescu and Tóth [4] for this problem; this modification improves the approximation ratio to 0.503. All the presented approximation algorithms run in $O(n \log n)$ time, where n is the number of input points.

We also study the MaxBPST problem in the special configurations of the input point set.

One special configuration is convex position. The $O(n^3)$-time algorithm of Borgelt *et al.* [3] for the MinBPST problem for points in convex position, also solves the MaxBPST problem for points in convex position within the same time bound. The semi-collinear configuration is another special case in which the red points are on a line and the blue points are on one side of the line. In Section 6 we present an $O(n^5)$-time algorithm that solves the MaxBPST problem for semi-collinear point sets. The presented algorithm, can also solve the MinBPST problem for this configuration within the same time bound. Concluding remarks are given in Section 7.

2 Preliminaries

For any two points p and q in the plane, we refer to the line segment between p and q as pq, and to the Euclidean distance between p and q as $|pq|$. The *lune* between p and q, which we denote by lune(p, q), is the intersection of the two disks of radius $|pq|$ that are centered at p and q.

For a point set P, the *diameter* of P is the maximum Euclidean distance between any two points of P. A pair of points that realizes the diameter of P is referred to as a *diametral pair.*

Let G be a geometric graph with colored vertices. We denote by $L(G)$ the total Euclidean length of the edges of G. A *star* is a tree with one internal node, which we refer to as the *center* of the star. For a color c, a *c-star* in G is a star whose center is colored c and the colors of its leaves are different from c.

3 The maximum bichromatic plane spanning tree problem

In this section we consider the MaxBPST problem. Recall that in this problem we are given two sets R and B of red and blue points in the plane, respectively, and we are looking for a maximum plane spanning tree in $K(R, B)$. Let $n = |R \cup B|$. We present an approximation algorithm with ratio 1/4 for this problem that runs in $O(n \log n)$ time.

We will show that the length of the longest star in $K(R, B)$ is at least 1/4 times the length of an optimal MaxBPST. In fact, we present an algorithm that returns such a star. Moreover, we show that this estimate is the best possible for the length of a longest star. The longest star can easily be augmented to form a plane spanning tree as follows. The longest star has exactly one point of one color as its center, and all points of other color as its leaves. The edges of this star can be extended to partition the plane into convex cones, possibly except one cone; we split this cone into two convex cones by adding its bisector. Then, we connect all the remaining points in each cone to one of the leaves that is on the boundary of that cone.

If $|R| = 1$ or $|B| = 1$, then the problem is trivial. Assume $|R| \geqslant 2$ and $|B| \geqslant 2$. Our algorithm first computes a diametral pair (a, b) in R and a diametral pair (p, q) in B. Then, returns the longest star S_x in $K(R, B)$ that is centered at a point $x \in \{a, b, p, q\}$. Since diametral pairs of R and B can be computed in $O(n \log n)$ time, the running time of the algorithm follows. In the rest of this section we will show that the longest of these stars satisfies the approximation ratio.

Let T^* be an optimal MaxBPST and let L^* denote the length of T^*. We make an arbitrary point the root of T^* and partition the edges of T^* into two sets as follows. Let E^*_R be the set of edges (u, v) in T^* where u is a red point and u is the parent of v. Let E^*_B be the set of edges (u, v) where u is a blue point and u is the parent of v. The edges of E^*_R (resp. E^*_B) form a forest in which each component is a red-star (resp. blue-star). Let L^*_R and L^*_B denote the total lengths of the edges of E^*_R and E^*_B, respectively. Without loss of generality assume that $L^*_R \leqslant L^*_B$. Then,

$$L^* = L^*_B + L^*_R \leqslant 2L^*_B. \tag{1}$$

We will show that, in this case, the longest of S_p and S_q is a desired tree. To that end, let F_B be the set of edges that is obtained by connecting every red point to its farthest blue point. Notice that the edges of F_B form a forest in which every component is a blue-star. Moreover, observe that

$$L_B^* \leqslant L(F_B). \tag{2}$$

Lemma 1. $L(F_B) \leqslant \frac{2}{\sqrt{3}} \cdot (L(S_p) + L(S_q))$.

Proof. Let C_p and C_q be the two disks of radius $|pq|$ that are centered at p and q, respectively. Since (p, q) is a diameter of B, all blue points lie in lune(p, q); see Figure 2(a).

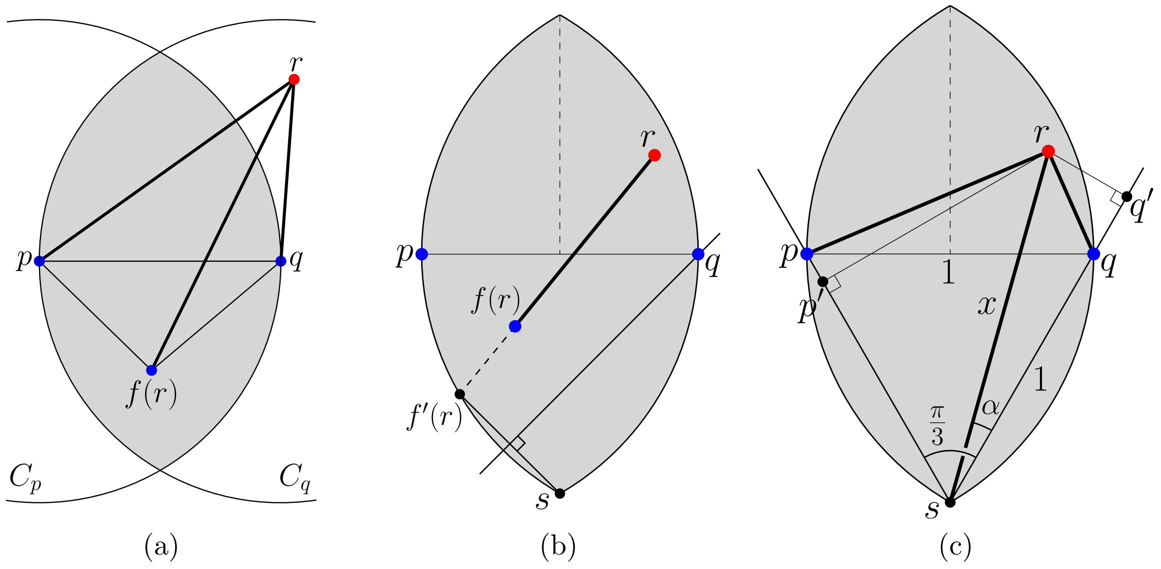


Figure 2: Illustration of Lemmas 1 and 2.

For any red point $r \in R$, let $f(r)$ denote its neighbor in F_B. Recall that $f(r)$ is the farthest blue point to r, and note that $f(r)$ is in lune(p, q). See Figure 2(a). We are going to show that $|rf(r)| \leqslant \frac{2}{\sqrt{3}}(|rp| + |rq|)$. Depending on whether or not $r \in$ lune(p, q) we consider the following two cases.

- $r \notin$ lune(p, q). Thus, we have that $r \notin C_p$ or $r \notin C_q$. Without loss of generality assume $r \notin C_p$; see Figure 2(a). By the triangle inequality we have $|rf(r)| \leqslant |rq| + |qf(r)|$. Since pq is a diameter of B, we have $|qf(r)| \leqslant |pq|$. In addition, since $r \notin C_p$, we have $|pq| \leqslant |rp|$. By combining these inequalities, we get

$$|rf(r)| \leqslant |rq| + |qf(r)| \leqslant |rq| + |pq| \leqslant |rq| + |rp|.$$

- $r \in$ lune(p, q). Without loss of generality assume that pq has unit length, pq is horizontal, r is above pq, and r is closer to q than to p. If $f(r)$ is on or above pq, then $|rf(r)|$ is smaller than $|pq|$, and hence smaller than $|rp| + |rq|$. Assume $f(r)$ is below pq. Let s be the intersection point of the boundaries of C_p and C_q that is below pq as in Figure 2(b).

  Claim 1. $|rf(r)| \leqslant |rs|$.

  Let $f'(r)$ be the intersection point of the ray that is emanating from r and passing through $f(r)$ with the boundary of lune(p, q). Note that $|rf(r)| \leqslant |rf'(r)|$. If $f'(r)$ is on the boundary of C_q, then the perpendicular bisector of the segment $sf'(r)$ passes through q. In this case r is in the same side of this perpendicular as $f'(r)$, and thus, $|rf'(r)| \leqslant |rs|$;

see Figure 2(b). Similarly, if $f'(r)$ is on the boundary of C_p, then both r and $f'(r)$ are on a same side of the perpendicular bisector of $sf'(r)$ which passes through p. This proves the claim.

Extend the line segment sp from the endpoint p. Let p' be the point on the extended line that is closest to r. Define q' similarly. Note that $|rp'| \leqslant |rp|$ and $|rq'| \leqslant |rq|$. Based on this and Claim 1, in order to show that $|rf(r)| \leqslant \frac{2}{\sqrt{3}}(|rp| + |rq|)$, it suffices to show that $|rs| \leqslant \frac{2}{\sqrt{3}}(|rp'| + |rq'|)$. Let $\alpha = \angle rsq$, and note that $0 \leqslant \alpha \leqslant \frac{\pi}{6}$; see Figure 2(c). Since the triangles $\triangle rsp'$ and $\triangle rsq'$ are right-angled and $\triangle spq$ is equilateral, we have $|rq'| = |rs| \cdot \sin\alpha$ and $|rp'| = |rs| \cdot \sin(\pi/3 - \alpha)$. Thus,

$$|rq'| + |rp'| = |rs| \cdot (\sin\alpha + \sin(\pi/3 - \alpha)) \geqslant \tfrac{\sqrt{3}}{2}|rs|,$$

where the inequality is valid because $\sin\alpha + \sin(\pi/3 - \alpha)$ is at least $\sqrt{3}/2$ for all $0 \leqslant \alpha \leqslant \frac{\pi}{6}$. This implies that $|rs| \leqslant \frac{2}{\sqrt{3}}(|rp'| + |rq'|)$.

Since in both previous cases $|rf(r)| \leqslant \frac{2}{\sqrt{3}}(|rp| + |rq|)$, we have

$$L(F_B) = \sum_{r \in R} |rf(r)| \leqslant \sum_{r \in R} \frac{2}{\sqrt{3}}(|rp| + |rq|) = \frac{2}{\sqrt{3}}(L(S_p) + L(S_q)).$$

□

Combining Inequalities (1), (2), and Lemma 1, we get

$$L^* \leqslant 2L^*_B \leqslant 2L(F_B) \leqslant \frac{4}{\sqrt{3}} \cdot (L(S_p) + L(S_q)).$$

Therefore, the length of the longest of S_p and S_q is at least $\frac{\sqrt{3}}{8} \approx 0.215$ times L^*. In the following lemma we improve the bound of Lemma 1 by proving that $L(F_B) \leqslant L(S_p) + L(S_q)$; this improves the approximation ratio to 1/4. However, the proof of this lemma is algebraic.

Lemma 2. *$L(F_B) \leqslant L(S_p) + L(S_q)$.*

Proof. For any red point $r \in R$, let $f(r)$ denote its neighbor in F_B. In order to prove this lemma, as we have seen in the proof of Lemma 1, it suffices to show that $|rf(r)| \leqslant |rp| + |rq|$. Define lune$(p, q)$ as in the proof of Lemma 1. As we have seen there, if $r \notin$ lune(p, q), then $|rf(r)| \leqslant |rp| + |rq|$. Assume $r \in$ lune(p, q). Without loss of generality assume that pq has unit length, pq is horizontal, r is above pq, and r is closer to q than to p; see Figure 2(c). Define the point s as in the proof of Lemma 1. By Claim 1 in the proof of Lemma 1, in order to show that $|rf(r)| \leqslant |rp| + |rq|$, it suffices to show that $|rs| \leqslant |rp| + |rq|$. Let $x = |rs|$ and $\alpha = \angle rsq$; see Figure 2(c). Note that $\sqrt{3}/2 \leqslant x \leqslant \sqrt{3}$ and $0 \leqslant \alpha \leqslant \frac{\pi}{6}$. By the cosine rule we have

$$|rp| = \sqrt{1 + x^2 - 2x\cos(\pi/3 - \alpha)} \qquad \text{and} \qquad |rq| = \sqrt{1 + x^2 - 2x\cos\alpha}.$$

Define

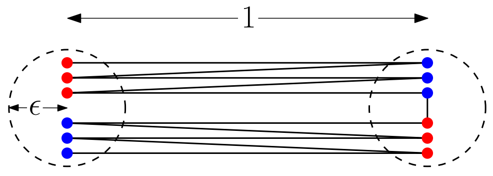
$$f(x, \alpha) = \sqrt{1 + x^2 - 2x\cos(\pi/3 - \alpha)} + \sqrt{1 + x^2 - 2x\cos\alpha} - x. \tag{3}$$

Then, $|rp| + |rq| - |rs| = f(x, \alpha)$. In Appendix A we show that $f(x, \alpha) \geqslant 0$ for all $\sqrt{3}/2 \leqslant x \leqslant \sqrt{3}$ and $0 \leqslant \alpha \leqslant \frac{\pi}{6}$. This implies that $|rs| \leqslant |rp| + |rq|$. □

To this end, we have proved the following theorem:

Theorem 1. *Let R and B be two disjoint sets of points in the plane such that $R \cup B$ is in general position, and let $n = |R \cup B|$. One can compute, in $O(n \log n)$ time, a plane spanning tree in $K(R, B)$ whose length is at least $1/4$ times the length of a maximum plane spanning tree.*

3.1 A matching upper bound



In this section we show that the above estimate is best possible for the length of the longest star in $K(R,B)$. Consider a set R of $n/2$ red points and a set B of $n/2$ blue points that are equally distributed in two circles of arbitrary very small radius with their centers at distance 1; see the figure to the right. The bichromatic plane spanning tree/path that is shown in this figure, has $n-2$ edges of unit length and one small edge. Any star in $K(R,B)$ has $n/4$ edges of unit length, plus $n/4$ edges of very small length. Thus, the length of the longest star, in this example, is about $1/4$ times the length of an optimal MaxBPST (at the limit).

4 The maximum k-colored plane spanning tree problem

In the multicolored version of the maximum plane spanning tree problem, the input points are colored by more than two colors, and we want the two endpoints of every edge in the tree to have distinct colors. Formally, we are given a set P of n points in the plane in general position that is partitioned into subsets $P_1,\dots,P_k$, with $k \geqslant 3$. For each $c \in \{1,\dots,k\}$, assume the points of P_c are colored c. Let $K(P_1,\dots,P_k)$ be the complete multipartite geometric graph on P, which has edges between every point of each set in the partition to all points of the other sets. The Max-k-PST problem is the problem of computing a maximum plane spanning tree in $K(P_1,\dots,P_k)$. The standard MaxPST problem can be interpreted as an instance of this multicolored version in which $k=n$, i.e., each point has a unique color. In this section, we present an approximation algorithm, for the Max-k-PST problem, whose ratio is $1/6$ if $k=3$, and $1/8$ if $k \geqslant 4$.

We will show that the length of the longest star in $K(P_1,\dots,P_k)$ is at least $1/8$ (resp. $1/6$) times the length of an optimal Max-k-PST if $k \geqslant 4$ (resp. $k=3$). In fact, we present an $O(n\log n)$-time algorithm that returns such a star. The algorithm is as follows. Compute a *bichromatic diameter* (p,q) of P, i.e., two points of different colors that have the maximum distance. It can easily be verified, by contradiction, that the MaxST in $K(P)$, which can be computed in $O(n\log n)$ time [5], contains a bichromatic diameter of P. Notice that the length of any edge in $K(P_1,\dots,P_k)$ is at most $|pq|$. Without loss of generality assume that $p \in P_i$ and $q \in P_j$. Notice that all points of $P \setminus (P_i \cup P_j)$ lie in lune(p,q), because, otherwise (p,q) cannot be a bichromatic diameter of P. To simplify the notation, we write R, B, and G, for P_i, P_j, and $P \setminus (P_i \cup P_j)$, respectively. Moreover, we assume that the points of R, B, and G are colored red, blue, and green, respectively. Compute a diametral pair (r,r') in R, a diametral pair (b,b') in B, and a diametral pair (g,g') in G. Return the longest star S_x in $K(P_1,\dots,P_k)$ that is centered at a point $x \in \{p,q,r,r',b,b',g,g'\}$. We show that the length of S_x is at least $1/8$ times the length of an optimal tree.

Let T^* be an optimal Max-k-PST, and let L^* denote the length of T^*. We make an arbitrary point the root of T^* and partition the edges of T^* into four sets as follows. Let E^*_R be the set of edges (u,v) in T^* where u is a red point and u is the parent of v. Let E^*_B be the set of edges (u,v) where u is a blue point and u is the parent of v. Let E^*_G be the set of edges (u,v) where u is a green point, v is not a green point, and u is the parent of v. Let E^* be the set of edges (u,v) where both u and v are green points. The edges of each of E^*_R, E^*_B, and E^*_G form a forest in which each component is a red-star, a blue-star, and a green-star, respectively. Let L^*_R, L^*_B, and L^*_G denote the total lengths of the edges of E^*_R, E^*_B, and E^*_G, respectively. Let L^*_E denote the total length of the edges in E^*. Then,

$$L^* = L^*_R + L^*_B + L^*_G + L^*_E.$$

We consider two cases depending on where or not L^*_E is larger than $\max\{L^*_R, L^*_B, L^*_G\}$.

- $L^*_E \geqslant \max\{L^*_R, L^*_B, L^*_G\}$. In this case $L^*_E \geqslant \frac{1}{4}L^*$. The number of edges in E^* is at most $n(G) - 1$, where $n(G)$ is the number of points in G. Recall that the length of every edge in E^* is at most $|pq|$. Thus $L^*_E \leqslant (n(G) - 1) \cdot |pq|$. Each of the stars S_p and S_q has an edge to every point of G. Thus $L(S_p) + L(S_q) \geqslant n(G) \cdot |pq|$. Therefore,

  $$\frac{1}{4}L^* \leqslant L^*_E \leqslant (n(G) - 1) \cdot |pq| < n(G) \cdot |pq| \leqslant L(S_p) + L(S_q),$$

  which implies that the longest of S_p and S_q has length at least $\frac{1}{8}L^*$.

- $L^*_E < \max\{L^*_R, L^*_B, L^*_G\}$. Without loss of generality assume that $L^*_B = \max\{L^*_R, L^*_B, L^*_G\}$. Thus, $L^*_B \geqslant \frac{1}{4}L^*$. Let F_B be the set of edges that is obtained by connecting every point of $R \cup G$ to its farthest blue point. Notice that the edges of F_B form a forest in which every component is a blue-star. Observe that $L^*_B \leqslant L(F_B)$. Moreover, by Lemma 2 we have $L(F_B) \leqslant L(S_b) + L(S_{b'})$. Therefore,

  $$\frac{1}{4}L^* \leqslant L^*_B \leqslant L(F_B) \leqslant L(S_b) + L(S_{b'}),$$

  which implies that the longest of S_b and $S_{b'}$ has length at least $\frac{1}{8}L^*$.

The point set $\{p, q, r, r', b, b', g, g'\}$ can be computed in $O(n \log n)$ time, and thus, the running time of the algorithm follows.

Note 1. If $k = 3$, then E^* is empty, and thus, the longest star S_x with $x \in \{r, r', b, b', g, g'\}$ has length at least $\frac{1}{6}L^*$.

Note 2. If the diameter pair of P is monochromatic, then we get the ratio of 1/6. Assume both points of a diametral pair (p, q) of P belong to P_i. Let $R = P_i$ and $G = R \setminus P_i$. Then, B is empty and $L^* = L^*_R + L^*_G + L^*_E$. Moreover, we have $r = p$ and $r' = q$. If $L^*_G \geqslant \frac{1}{3}L^*$, then the longest of S_g and $S_{g'}$ has length at least $\frac{1}{6}L^*$. If $L^*_G < \frac{1}{3}L^*$, then $L^*_R \leqslant L(S_r) + L(S_{r'})$ and $L^*_E \leqslant L(S_r) + L(S_{r'})$. Thus, the longest of S_r and $S_{r'}$ has length at least $\frac{1}{6}L^*$.

Theorem 2. *Let P be a set of n points in the plane in general position that is partitioned into subsets $P_1, \dots, P_k$, with $k \geqslant 3$. One can compute, in $O(n \log n)$ time, a plane spanning tree in $K(P_1, \dots, P_k)$ whose length is at least* 1/8 (*resp.* 1/6) *times the length of a maximum plane spanning tree if $k \geqslant 4$* (*resp.* $k = 3$).

5 The maximum plane spanning tree problem

In this section we study the MaxPST problem, a special case of the Max-k-PST problem where every input point has a unique color. Formally, given a set P of points in the plane in general position and we want to compute a maximum plane spanning tree in $K(P)$. We revisit this problem which was first studied by Alon, Rajagopalan and Suri (1993), and then by Dumitrescu and Tóth (2010). Alon *et al.* [1] presented an approximation algorithm with ratio 1/2 for this problem. In fact they proved that the length of the longest star in $K(P)$ is at least 0.5 times the length of an optimal tree; this bound is tight for a longest star. Dumitrescu and Tóth [4] improved the approximation ratio to 0.502. They proved that the length of the longest double-star in $K(P)$ (a double-star is a tree with two internal nodes) is at least 0.502 times the length of an optimal solution. They left as an open problem a more precise analysis of the approximation ratio of their algorithm. In this section we modify the algorithm of Dumitrescu and Tóth [4], and slightly improve the approximation ratio to 0.503. We will describe their algorithm briefly, and provide detail on the parts that we will modify. The algorithm outputs the longest of five plane trees S_a, S_b, S_h, E_a, E_b, that are describe below.

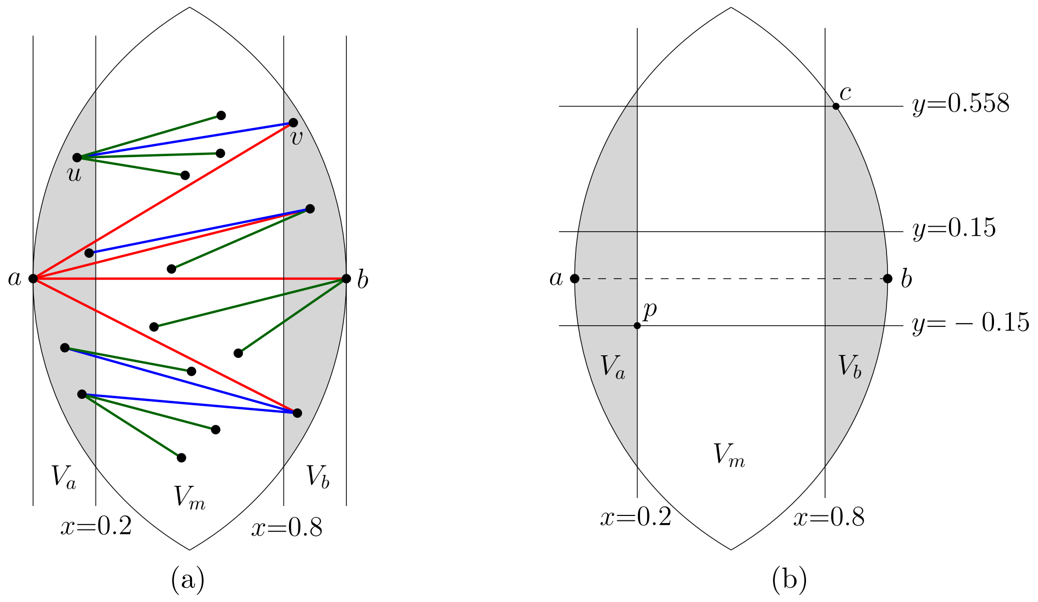


Figure 3: (a) The plane tree E_a. The red, blue, and green edges belong to E_a^1, E_a^2, and E_a^3 respectively. (b) The distance between p and c is less than 0.948.

Let $n = |P|$, and let L^* denote the length of an optimal MaxPST in $K(P)$. Compute a diametral pair (a, b) of P. Without loss of generality assume that ab is a horizontal, $|ab| = 1$, $a = (0, 0)$, and $b = (1, 0)$. Since (a, b) is a diametral pair, the length of any edge in $K(P)$ is at most 1. Thus, $L^* \leqslant n - 1$. Moreover, all points of P are in lune(a, b). See Figure 3. Let $h = (x_h, y_h)$ be a point in P with the largest value of $|y|$ (absolute value of the y-coordinate). Without loss of generality assume that $y_h \geqslant 0$. Define S_a, S_b, and S_h as three spanning stars that are centered at a, b, and h respectively. We compute plane trees E_a and E_b as follows; this computation is different from the one that is presented in [4].

Set $w = 0.6$. Let V_a be the vertical strip between the lines $x = 0$ and $x = \frac{1-w}{2} = 0.2$, V_m be the vertical strip between the lines $x = 0.2$ and $x = \frac{1+w}{2} = 0.8$, and V_b be the vertical strip between the lines $x = 0.8$ and $x = 1$. Note that $a \in V_a$ and $b \in V_b$. See Figure 3(a). We describe how to construct E_a; the construction of E_b is analogous. Connect a to each point in V_b and let E_a^1 denotes the set of edges of the resulting star (the red edges in Figure 3(a)). The edge ab has length 1 and every other edge has length at least $\frac{1+w}{2}$. The edges of E_a^1 partition V_a into convex regions. Each of these regions is bounded by at least one edge of E_a^1 from above or below. Take any region $\mathcal{A}$ of the partition of V_a. Let av be an edge of E_a^1 that bounds $\mathcal{A}$ either from above or blow. Note that $v \in V_b$. Connect all points that are in $\mathcal{A}$ (excluding a) to v. Let E_a^2 be the set of all such edges after considering all regions (the blue edges in Figure 3(a)). Since any edge in E_a^2 connects a point in V_a to a point in V_b, each edge in E_a^2 has length at least w. The edges of $E_a^1 \cup E_a^2$ partition V_m into convex regions. Each of these regions is bounded by at least one edge of $E_a^1 \cup E_a^2$. Take any region $\mathcal{M}$ of the partition of V_m. Consider the following two cases: (a) If $\mathcal{M}$ is bounded by an edge uv of E_a^2, then connect all points that are in $\mathcal{M}$ to one of u and v that maximizes the total length of the new edges, and (b) if $\mathcal{M}$ is not bounded by any edge of E_a^2, then it is bounded by an edge av of E_a^1, connect all points that are in $\mathcal{M}$ to one of a and v that maximizes the total length of the new edges. Let E_a^3 be the set of all such edges after considering all regions of V_m (the green edges in Figure 3(a)). Let E_a be the graph with vertex set P and edge set $E_a^1 \cup E_b^2 \cup E_a^3$.

The following is a restatement of Lemma 3 in [4].

Lemma 3 (Dumitrescu and Tóth [4]). *Let a and b be two points in the plane that are at distance at least α from each other, for some real $\alpha > 0$. Let S be a set of m points in the plane. Let S_a and S_b be two star that are centered at a and b, respectively, and connected to all points of S. Then, the length of the longest of S_a and S_b is at least $\frac{\alpha \cdot m}{2}$.*

Recall that the length of each edge in E_a^1 is at least $\frac{1+w}{2}$ and the length of each edge in E_b^2 is at least w. By Lemma 3 the total length of the edges in E_a^3 is at least $\frac{w}{2}$ times the number of points in V_m.

Lemma 4. *Let n_a and n_b denote the number of points in V_a and V_b respectively. Then $L(E_a) \geqslant \frac{n_b}{2} + \frac{w}{2}(n + n_a) + \frac{1-3w}{2}$, $L(E_b) \geqslant \frac{n_a}{2} + \frac{w}{2}(n + n_b) + \frac{1-3w}{2}$, and consequently $L(E_a) + L(E_b) \geqslant \frac{n_a+n_b}{2} + \frac{w}{2}(2n + n_a + n_b) + 1 - 3w$.*

Proof. Since E_a^1, E_a^2, and E_a^3 are pairwise disjoint, we have $L(E_a) = L(E_a^1) + L(E_a^2) + L(E_a^3)$. E_a^1 contains n_b edges (including ab) of length at least $\frac{1+w}{2}$ with the length of ab is 1. Thus, $L(E_a^1) \geqslant \frac{1+w}{2}(n_b - 1) + 1$. E_a^2 has $n_a - 1$ edges of length at least w. Thus, $L(E_a^2) \geqslant w(n_a - 1)$. V_m contains $n - n_a - n_b$ points, and thus, the length of E_a^3 is at least $\frac{w}{2}(n - n_a - n_b)$. Therefore,

$$L(E_a) \geqslant \frac{1+w}{2}(n_b - 1) + 1 + w(n_a - 1) + \frac{w}{2}(n - n_a - n_b) = \frac{n_b}{2} + \frac{w}{2}(n + n_a) + \frac{1-3w}{2}.$$

The estimation of $L(E_b)$ is analogous. □

The following lemma summarizes Lemmas 3, 4, 5, and 7 in [4]. Let $P = \{p_1, \dots, p_n\}$, where $p_i = (x_i, y_i)$. Let $d_{max}(p_i)$ denote the maximum distance from p_i to other points in P.

Lemma 5 (Dumitrescu and Tóth [4]). *Let δ and t be two constants where $0 \leqslant \delta \leqslant t \leqslant 1$. Then*

1. *$L(S_a) + L(S_b) \geqslant n$.*
2. *$L^* \leqslant \sum_{i=1}^n d_{max}(p_i)$.*
3. *if $\sum_{i=1}^n |y_i| \geqslant \delta n$, then $L(S_a) + L(S_b) \geqslant 2n\sqrt{\frac{1}{4} + \delta^2}$.*
4. *if $\sum_{i=1}^n |y_i| \leqslant \delta n$ and $y_h \geqslant t$, then $L(S_h) \geqslant (t - \delta)n$.*

Set $\delta = 0.055$, $t = 0.558$, and $z = 0.49$; these constants are different from the ones that are chosen in [4].

Lemma 6. *Assume that $|y_h| \leqslant t$. Let $p_i = (x_i, y_i)$ be a point of P in V_m with $|y_i| \leqslant 0.15$. Then $d_{max}(p_i) \leqslant 0.948$.*

Proof. The maximum distance is attained when $p_i = (0.2, -0.15)$ or $p_i = (0.8, -0.15)$. Because of symmetry we assume that $p_i = (0.2, -0.15)$. Let $c = (x_c, y_c)$ be the rightmost intersection point of the line $y = t$ and lune(a, b). See Figure 3(b). The furthest point from p_i in the allowed region is c. Note that $y_c = t$ and $x_c = \sqrt{1 - t^2} < 0.83$. Thus, $x_c - x_i < 0.63$. Therefore $d_{max}(p_i) \leqslant |p_i c| < \sqrt{0.63^2 + (0.15 + t)^2} < 0.948$. □

To prove the approximation ratio, we show that the length of the longest of S_a, S_b, S_h, E_a, E_b is at least 0.503 times L^*. In order to do that, we recall the following four cases that are considered in [4].

1. If $\sum_{i=1}^{n} |y_i| \geqslant \delta n$, then the output of the algorithm is not shorter than the longest of S_a and S_b. By Lemma 5, the approximation ratio is at least
$$\frac{L(S_a) + L(S_b)}{2L^*} \geqslant \sqrt{\frac{1}{4} + \delta^2} \geqslant 0.503.$$

2. If $\sum_{i=1}^{n} |y_i| < \delta n$ and $y_h \geqslant t$, then the output of the algorithm is not shorter than S_h. By Lemma 5, the approximation ratio is at least
$$\frac{L(S_h)}{L^*} \geqslant t - \delta = 0.503.$$

3. If $\sum_{i=1}^{n} |y_i| < \delta n$, $y_h < t$, and $n_a + n_b \geqslant (1-z)n$, then the output of the algorithm is not shorter than the longest of E_a and E_b. As a consequence of Lemma 4, the approximation ratio is at least
$$\begin{aligned}\frac{L(E_a) + L(E_b)}{2L^*} &\geqslant \frac{n_a + n_b}{2 \cdot 2n} + \frac{w(2n + n_a + n_b)}{2 \cdot 2n} + \frac{1 - 3w}{2n} \\ &\geqslant \frac{(1-z)(1+w)}{4} + \frac{w}{2} + \frac{1-3w}{2n} \geqslant 0.503,\end{aligned}$$
where the last inequality is valid for all $n \geqslant 400$. Based on the above calculations and the details that are provided in the proof of Lemma 4, it turns out that ratio of 0.502 that is claimed in [4] is valid when $n \geqslant 100$.

4. If $\sum_{i=1}^{n} |y_i| < \delta n$, $y_h < t$, and $n_a + n_b < (1-z)n$, then the output of the algorithm is not shorter than the longest of S_a and S_b. There are at least $zn = .49n$ points in V_m. At most $\frac{11}{30}$ points have $|y_i| \geqslant 0.15$ because otherwise we have $\sum_{i=1}^{n} |y_i| \geqslant 0.15 \cdot \frac{11n}{30} = 0.055n = \delta n$, which is a contradiction. Thus, at least $\frac{49n}{100} - \frac{11n}{30} = \frac{37n}{300}$ points in V_m have $|y_i| \leqslant 0.15$. By Lemma 5 and Lemma 6 we have
$$L^* \leqslant \frac{263n}{300} + 0.948 \cdot \frac{37n}{300} < 0.994n.$$
The approximation ratio is at least
$$\frac{L(S_a) + L(S_b)}{2L^*} \geqslant \frac{n}{2 \cdot 0.994n} > 0.503.$$

Theorem 3. *Let P be a set of n points in the plane in general position. One can compute, in $O(n \log n)$ time, a plane spanning tree in $K(P)$ whose length is at least 0.503 times the length of a maximum plane spanning tree.*

6 The MaxBPST problem on special point sets

In this section we consider the MaxBPST problem on some restricted point sets, such as

(*i*) point sets that are in convex position,

(*ii*) point sets that are semi-collinear, and

(*iii*) point sets with only two red points.

6.1 Points in convex position

Assume $R \cup B$ is in convex position and let $|R| \leqslant |B|$. Borgelt *et al.* [3] presented a dynamic programming algorithm that solves the MinBPST problem, in $K(R, B)$, in $O(|R|^2|B|)$ time. Their algorithm can be adjusted to solve the MaxBPST within the same time bound.

6.2 Semi-collinear point sets

In this section we consider the special case where the input points are semi-collinear, i.e., the red points lie on a line ℓ and the blue points lie on one side of ℓ. Without loss of generality, we assume that ℓ is horizontal and the blue points lie above ℓ. Let R and B denote the non-empty sets of red and blue points, respectively, and let $n = |R \cup B|$. We present a top-down dynamic programming algorithm that solves the MaxBPST problem for semi-collinear points in $O(n^5)$ time. In a dynamic programming algorithm we maintain a table that stores the solutions of subproblems. In a top-down dynamic programming algorithm, when we encounter a subproblem, first we check the corresponding table-entry; if the subproblem is already solved, then we use that solution, otherwise, we solve the subproblem recursively.

6.2.1 Subproblems

Let $m = |R|$, then $|B| = n - m$. Let $r_1, \dots, r_m$ be the sequence of red points from left to right, and let $\{b_1, \dots, b_{n-m}\}$ be the set of blue points. We introduce $A(i, j, u, v, u', v', w)$ to be the problem of computing the weight of a MaxBPST that spans a red point set R_A and a blue point set B_A where

- i and j, with $i \leqslant j$, are the indices of two red points such that $R_A = \{r_i, \dots, r_j\}$.
- u and v are the indices of two blue points such that B_A contains the blue points that are to the right side of $\ell(r_i, b_u)$, to the left side of $\ell(r_j, b_v)$, and lower than both b_u and b_v.
- u' (resp. v') takes a value in the set $\{\text{in}, \text{ex}\}$ which indicates whether the blue point b_u (resp. b_v) is included in B_A or excluded.
- w takes a value in the set $\{\text{con}, \text{nco}\}$ where con stands for "connected" and nco stands for "not connected".

The solution for $A(\cdot)$ satisfies the following two rules:

- Rule 1: If $u' = \text{in}$ (resp. $v' = \text{in}$), then we have a constrained version of $A(\cdot)$ where $b_u r_i$ (resp. $b_v r_j$) should be part of the solution, but $A(\cdot)$ returns the total weight of new edges, i.e., the weight of a MaxBPST minus $|b_u r_i|$ (resp. $|b_v r_j|$).
- Rule 2: If $u' = v' = \text{in}$, then $u = v$, and consequently $b_u = b_v$. Thus, every time we call $A(\cdot)$ with $u' = v' = \text{in}$, we ensure that $u = v$.

While maintaining the above two rules, the output of $A(\cdot)$ is determined as follows:

- if $w = \text{con}$, then the output is the weight of a MaxBPST tree on $R_A \cup B_A$.
- if $w = \text{nco}$ and $u' = v' = in$, then the output is the weight of a MaxBPST for $R_A \cup B_A$.
- if $w = \text{nco}$, then we find two vertex-disjoint bichromatic trees T_i and T_j that span $R_A \cup B_A$, where $T_i \cup T_j$ is plane, r_i is a vertex of T_i, r_j is a vertex of T_j, and the total weight of $T_i \cup T_j$ is maximized. Following Rule 1, if $u' = \text{in}$, then T_i should contain $r_i b_u$, and if $v' = \text{in}$, then T_j should contain $r_j b_v$. In addition, the output is the total weight of $T_i \cup T_j$ except the weight of the edges $r_i b_u$ and $r_j b_v$.

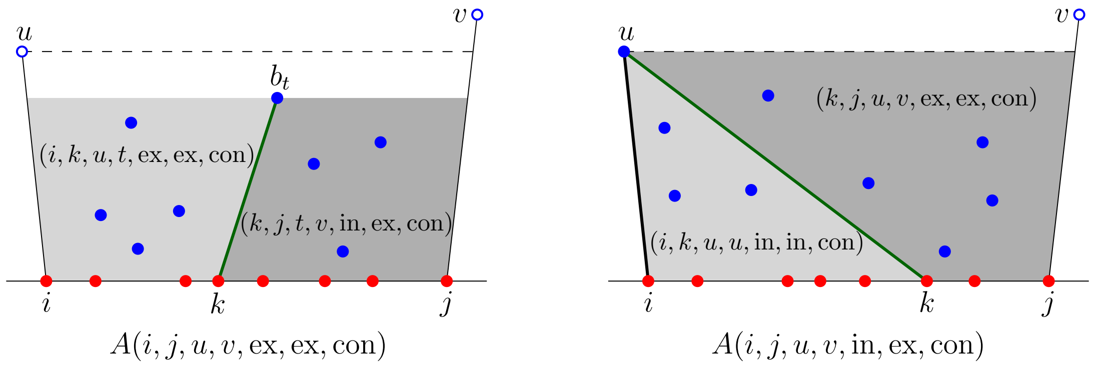
Depending on the values of u', v', and w, we have eight different configurations for $A(\cdot)$. Starting from our top-level call as $A(\cdot,\cdot,\cdot,\cdot,\text{ex},\text{ex},\text{con})$ (that is described in Subsection 6.2.2) we will never call three cases $A(i,j,u,v,\text{in},\text{in},\text{nco})$, $A(i,j,u,v,\text{ex},\text{in},\text{con})$ and $A(i,j,u,v,\text{ex},\text{in},\text{nco})$ during the recursive calls. So, we will describe how to handle each of the remaining five configurations (we will describe the first two configurations in detail and the remaining cases briefly). During recursive calls to $A(\cdot)$ we make sure that the size of input gets smaller, or we transfer from one configuration to another configuration which decreases the size of input. Therefore, the algorithm will terminate. In each of the configurations, if $|R_A| \leqslant 1$ or $|B_A| \leqslant 1$, then the solution is trivial. Thus, in the description of the following configurations we assume that $|R_A| \geqslant 2$ and $|B_A| \geqslant 2$.

1. *Subproblem* $A(i,j,u,v,\text{ex},\text{ex},\text{con})$

Since $u' = \text{ex}$ and $v' = \text{ex}$, the set B_A does not contain any of b_u and b_v. Since $w = \text{con}$, $A(\cdot)$ returns the wight of a maximum bichromatic plane tree that spans $R_A \cup B_A$. Let b_t be a topmost point in B_A. In any solution of $A(\cdot)$, b_t is connected to at least one red point; let r_k, with $k \in \{i,\dots,j\}$, be the leftmost such point. Since b_t is a topmost blue point, the solution does not contain an edge e that connects a blue point to the left (resp. right) of $\ell(b_t, r_k)$ to a red point to the right (resp. left) of $\ell(b_t, r_k)$, because, otherwise, e crosses $b_t r_k$. Thus, the edge $b_t r_k$ splits the problem into two subproblems, one to the left and one to to the right. See Figure 4(left). Since r_k is the leftmost red point that b_t is connected to, b_t is not an input point to the left subproblem (b_t is excluded). However, b_t might be connected to some red points to the right of r_k, and thus, b_t is an input point to the right subproblem (b_t is included). Therefore, we have

$$A(i,j,u,v,\text{ex},\text{ex},\text{con}) = |b_t r_k| + A(i,k,u,t,\text{ex},\text{ex},\text{con}) + A(k,j,t,v,\text{in},\text{ex},\text{con}).$$

By Rule 1, the value that is returned by $A(k,j,t,v,\text{in},\text{ex},\text{con})$ does not include $|b_t r_k|$. This makes sure that, in $A(i,j,u,v,\text{ex},\text{ex},\text{con})$, we do not count $|b_t r_k|$ twice. Since we do not know the value of k, we try all $j-i+1$ possible values for k, then pick the one that maximizes $A(i,j,u,v,\text{ex},\text{ex},\text{con})$.



u
v
b_t
$(i,k,u,t,\text{ex},\text{ex},\text{con})$
$(k,j,t,v,\text{in},\text{ex},\text{con})$
i
k
j
$A(i,j,u,v,\text{ex},\text{ex},\text{con})$
u
v
$(k,j,u,v,\text{ex},\text{ex},\text{con})$
$(i,k,u,u,\text{in},\text{in},\text{con})$
i
k
j
$A(i,j,u,v,\text{in},\text{ex},\text{con})$

Figure 4: Illustration for solving subproblems $A(i,j,u,v,\text{ex},\text{ex},\text{con})$ and $A(i,j,u,v,\text{in},\text{ex},\text{con})$.

2. *Subproblem* $A(i,j,u,v,\text{in},\text{ex},\text{con})$

In this subproblem the set B_A contains b_u but does not contain b_v. Moreover, the solution associated with this problem should contain the edge $b_u r_i$ while the value that is returned by $A(\cdot)$ does not contain $|b_u r_i|$. See Figure 4(right). We consider two cases depending on whether or not b_u is connected to any red point other than r_i. Then, we take the case that gives the maximum value. If b_u is not connected to any other red point, then

$$A(i, j, u, v, \text{in}, \text{ex}, \text{con}) = A(i, j, u, v, \text{ex}, \text{ex}, \text{con}).$$

If b_u is connected to some red points other than r_i, then let r_k, with $k \in \{i+1, \ldots, j\}$, be the rightmost one. Since b_u is the topmost blue point in B_A, the edge $b_u r_k$ splits the problem into two subproblems, one to the left and one to the right. Since r_k is the rightmost red point that b_u is connected to, b_u is not an input point to the right subproblem. The solution of the left subproblem is a tree that contains both edges $b_u r_i$ and $b_u r_k$. See Figure 4(right). Therefore, we have

$$A(i, j, u, v, \text{in}, \text{ex}, \text{con}) = |b_u r_k| + A(i, k, u, u, \text{in}, \text{in}, \text{con}) + A(k, j, u, v, \text{ex}, \text{ex}, \text{con}).$$

Notice that Rule 2 is maintained here. Also, by Rule 1, the value that is returned by $A(i, k, u, u, \text{in}, \text{in}, \text{con})$ does not include $|b_u r_i|$ nor $|b_u r_k|$. Since we do not know the value of k, we try all $j - i$ possible values of k, then pick the one that maximizes $A(\cdot)$.

3. *Subproblem* $A(i, j, u, u, \text{in}, \text{in}, \text{con})$

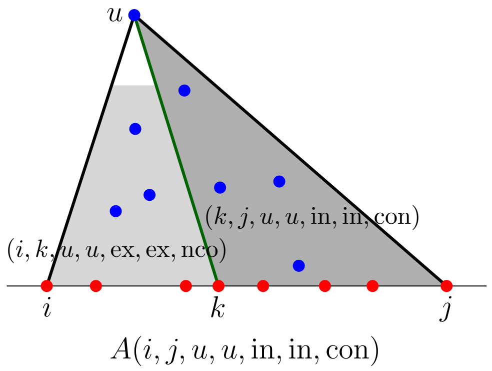
In this case $u = v$, and B_A contains b_u. Since $w = \text{in}$, the weight of a maximum bichromatic plane spanning tree on $R_A \cup B_A$. Since $u' = v' = \text{in}$, both edges $b_u r_i$ and $b_u r_j$ are in this tree, but, by Rule 1, $A(\cdot)$ returns the wight of this tree except these two edges. See Figure 5(left). We consider two cases depending on whether or not b_u is connected to any red point other than r_i and r_j. Then, we take the case that gives the maximum value. If b_u is not connected to any other red point, then

$$A(i, j, u, u, \text{in}, \text{in}, \text{con}) = A(i, j, u, u, \text{ex}, \text{ex}, \text{nco}).$$

If b_u is connected to some red points other than r_i and r_j, then let r_k, $k \in \{i+1, \ldots, j-1\}$, be the leftmost one; see Figure 5(left). This splits the problem into two subproblems such that

$$A(i, j, u, u, \text{in}, \text{in}, \text{con}) = |b_u r_k| + A(i, k, u, u, \text{ex}, \text{ex}, \text{nco}) + A(k, j, u, u, \text{in}, \text{in}, \text{con}).$$

We try all $j - i - 1$ possible values of k, then pick the one that maximizes $A(\cdot)$.


u
$(i, k, u, u, \text{ex}, \text{ex}, \text{nco})$
$(k, j, u, u, \text{in}, \text{in}, \text{con})$
i
k
j
$A(i, j, u, u, \text{in}, \text{in}, \text{con})$

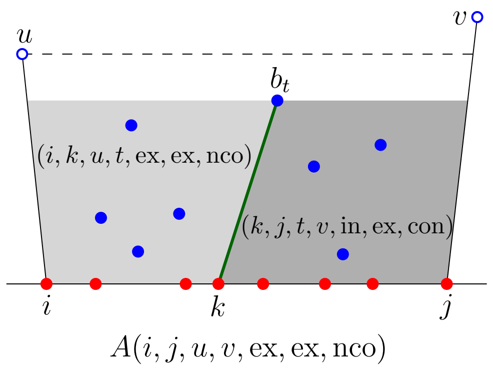

u
v
b_t
$(i, k, u, t, \text{ex}, \text{ex}, \text{nco})$
$(k, j, t, v, \text{in}, \text{ex}, \text{con})$
i
k
j
$A(i, j, u, v, \text{ex}, \text{ex}, \text{nco})$

Figure 5: Illustration for solving subproblems $A(i, j, u, u, \text{in}, \text{in}, \text{con})$ and $A(i, j, u, v, \text{ex}, \text{ex}, \text{nco})$.

4. *Subproblem* $A(i, j, u, v, \text{ex}, \text{ex}, \text{nco})$

In this subproblem, B_A does not contain any of b_u and b_v. The solution to this subproblem consists of two vertex disjoint spanning trees that span $R_A \cup B_A$, their union is plane, and r_i and

r_j are not in a same tree. Let b_t be a topmost point in B_A. In any solution of $A(\cdot)$, b_t is connected to at least one red point; let r_k, $k \in \{i, \ldots, j\}$, be the leftmost one. See Figure 5(right). We consider two cases: (1) $k \in \{i, j\}$, and (2) $k \in \{i+1, j-1\}$. Then, we take the one that maximizes $A(\cdot)$. In case (1) if $k = i$ then

$$A(i, j, u, v, \text{ex}, \text{ex}, \text{nco}) = |b_t r_i| + A(i, i, u, t, \text{ex}, \text{ex}, \text{con}) + A(i, j, t, v, \text{in}, \text{ex}, \text{nco}),$$

and if $k = j$ then

$$A(i, j, u, v, \text{ex}, \text{ex}, \text{nco}) = |b_t r_j| + A(i, j, u, t, \text{ex}, \text{ex}, \text{nco}) + A(j, j, t, v, \text{in}, \text{ex}, \text{con}).$$

In case (2) we have two subproblems, only in one of them r_k should not be connected to the other extreme red point. This raises two cases, thus, we take the one that gives a larger value.

$$\begin{aligned} A(i, j, u, v, \text{ex}, \text{ex}, \text{nco}) = |b_t r_k| + \max\{&A(i, k, u, t, \text{ex}, \text{ex}, \text{con}) + A(k, j, t, v, \text{in}, \text{ex}, \text{nco}), \\ &A(i, k, u, t, \text{ex}, \text{ex}, \text{nco}) + A(k, j, t, v, \text{in}, \text{ex}, \text{con})\}. \end{aligned}$$

We try all possible values of k, then pick the one that maximizes $A(\cdot)$.

5. *Subproblem* $A(i, j, u, v, \text{in}, \text{ex}, \text{nco})$

The optimal solution associated with this subproblem consists of two trees, one of them contains the edge $b_u r_i$. We consider two cases depending on whether or not b_u is connected any red point other than r_i, then take the one that maximizes $A(\cdot)$. If b_u is not connected to any other red point, then

$$A(i, j, u, v, \text{in}, \text{ex}, \text{nco}) = A(i, j, u, v, \text{ex}, \text{ex}, \text{nco}).$$

If b_u is connected to some red points other than r_i, then let r_k, $k \in \{i+1, \ldots, j-1\}$, be the rightmost one (note that k cannot be j, because, otherwise, the disconnectivity of r_i and r_j will not be maintained). This splits the problem into two subproblems. In the left subproblem, r_k is connected to r_i via b_u, thus, the disconnectivity of r_i and r_j should be maintained by the right subproblem. Therefore, we have

$$A(i, j, u, v, \text{in}, \text{ex}, \text{nco}) = |b_u r_k| + A(i, k, u, u, \text{in}, \text{in}, \text{con}) + A(k, j, u, v, \text{ex}, \text{ex}, \text{nco}).$$

Notice that both Rule 1 and Rule 2 are satisfied here. We try all $j - i - 1$ possible values of k, then pick the one that maximizes $A(\cdot)$.

6.2.2 The top-level problem

In this section we describe how to use problem $A(\cdot)$, that is defined in the previous section, to solve the original top level problem which is to compute a MaxBPST for semi-collinear points. Let $B = \{b_1, \ldots, b_{n-m}\}$ be the set of blue points and $r_1, \ldots, r_m$ be the sequence of red points from left to right. Add two (fake) blue points b_0 and b_{n-m+1} such that all points of B are to the right side of $\ell(r_1, b_0)$, to the left side of $\ell(r_m, b_{n-m+1})$, and lower than both b_0 and b_{n-m+1}. We maintain a table A such that each entry $A[i, j, u, v, u', v', w]$ stores the weight of a MaxBPST associated with subproblem $A(i, j, u, v, u', v', w)$. We initialize all table entries by a negative value. After filling table A by our dynamic programming algorithm, the weight of an optimal MaxBPST is stored in $A[1, m, 0, n-m+1, \text{ex}, \text{ex}, \text{con}]$; the tree itself can also be retrieved from table A.

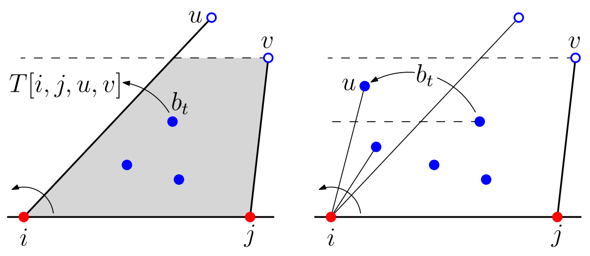
6.2.3 Running time analysis

The indices i and j go over the red points, while the indices u and v go over the blue points. Each of the indices u', v', and w takes only two values. Thus, we have $O(|R|^2|B|^2)$ subproblems in total, and hence the table A has $O(|R|^2|B|^2)$ entries. The blue points associated with each subproblem, and a topmost one, can be computed in $O(|B|)$ preprocessing time. Thus the total preprocessing time is $O(|R|^2|B|^3)$. After that, during the algorithm, to solve each subproblem, the index k goes over $O(|R|)$ red points. Thus, after preprocessing, we spend $O(|R|^3|B|^2)$ to solve all the subproblems. Therefore, the total running time of our algorithm is $O(|R|^3|B|^2+|R|^2|B|^3)$.

Notice that to solve each subproblem we do not need all the blue points that are associated with that subproblem, but we do need a topmost one. Moreover, notice that we need to find a topmost blue point only for subproblems of type $A(\cdot,\cdot,\cdot,\cdot,\text{ex},\text{ex},\cdot)$. We describe how to compute topmost blue points for all such subproblems in $O(|R|^2|B|^2)$ preprocessing time. This will improve the running time of our algorithm to $O(|R|^3|B|^2)$.

For simplicity of description, we assume that no two blue points have a same y-coordinate, however, for the purpose of our algorithm we do not need this assumption. The topmost blue point for a subproblem $A(i,j,u,v,\text{ex},\text{ex},w)$ is uniquely defined by i, j, u, and v; it is independent of w. Let $B = \{b_1,\dots,b_{|B|}\}$ denote the set of all blue points, including the two fake blue points that we add in the top level of the algorithm. Assume that points of R are on x-axis, and points of B lie above x-axis. Let $r_1,\dots,r_{|R|}$ be the red points from left to right. For each $i \in \{1,\dots,|R|\}$, let L_i^{c} be the sorted list of points of B in clockwise order around r_i by starting from x-axis. Similarly, let L_i^{cc} be the sorted list of points of B in counterclockwise order around r_i.

We maintain a table T whose entry $T[i,j,u,v]$ (with $i,j \in \{1,\dots,|R|\}$ and $u,v \in \{1,\dots,|B|\}$) stores the topmost blue point for the subproblem $A(i,j,u,v,\text{ex},\text{ex},w)$. Recall that the blue input set to each subproblem $A(i,j,u,v,\text{ex},\text{ex},w)$ is to the right side of $\ell(b_u,r_i)$, to the left side of $\ell(b_v,r_j)$, and lower than both b_u and b_v.



b_u is above b_v $\qquad$ b_u is below b_v

For each pair (i,j), with $i,j \in \{1,\dots,|R|\}$, we fill the entries $T[i,j,\cdot,\cdot]$, in two phases, as follows. In the first phase we fill the entries related to the cases where b_u is above b_v, and in the second phase we fill the entries related to the cases where b_u is below b_v. We describe the first phase; the second phase is analogous. We iterate over $v \in \{1,\dots,|B|\}$. In the vth iteration we compute the topmost blue point for all $A(i,j,u,v,\text{ex},\text{ex},w)$ where $u \in \{1,\dots,|B|\}$ and b_u is above b_v. Let b_t be the current topmost blue point that is to the left side of $\ell(b_v,r_j)$ and below b_v. Traverse the points of L_i^{cc} from the beginning, and let b_u be the current point. We have two cases:

- b_u is above b_v. Set $T[i,j,u,v] = b_t$, then proceed to the next point in L_i^{cc}.
- b_u is below b_v. If b_u is to the left of $\ell(b_v,r_j)$ and above b_t, then let b_u be the current b_t. Then proceed to the next point in L_i^{cc}.

This is the end of the first phase. In the second phase we iterate over $u \in \{1,\dots,|B|\}$, and in the uth iteration we traverse L_j^{c} and find $T[i,j,u,v]$ for b_v's that are above b_u. As for the running time, the lists L_i^{c} and L_i^{cc}, for all $i \in \{1,\dots,|R|\}$, can be computed in $O(|R||B|\log|B|)$ time. After that, we spend $O(|R|^2|B|^2)$ time to fill T. Therefore, our dynamic programming algorithm runs in $O(|R|^3|B|^2)$ time.

Theorem 4. *Let R and B be two disjoint sets of points in the plane such that the points of R lie on a straight line, the points of B are on one side of this line, and no two points of B are collinear with any point of R. Then, a maximum plane spanning tree in $K(R, B)$ can be computed in $O(|R|^3|B|^2)$ time.*

Note The dynamic programming algorithm presented in this section can easily be adjusted to solve the MinBPST problem, and also the bottleneck versions of the bichromatic plane spanning tree problem (minimizing the length of the longest edge, or maximizing the length of the shortest edge) for semi-collinear points in $O(|R|^3|B|^2)$ time.

6.3 Two red points

Borgelt *et al.* [3] showed that a minimum bichromatic plane spanning tree of two red points and n blue points can be computed in $O(n \log n)$ time. In this section we show how to compute a maximum bichromatic plane spanning tree for such a point set in $O(n^2)$ time. Let ℓ be the line passing through the two red points, and assume ℓ is horizontal. This introduces two instances of the semi-collinear case: one instance below ℓ and one instance above ℓ. In an optimal tree, either the two red points are connected by a blue point above ℓ and disconnected below ℓ, or the two red points are connected by a blue point below ℓ and disconnected above ℓ. Thus, we can compute an optimal tree by taking the longest of the two trees obtained from these two cases. Since $|R| = 2$, the running time of our algorithm for this case is $O(|R|^3|B|^2) = O(n^2)$.

7 Concluding Remarks

In this paper we presented constant factor approximation algorithms for the problem of computing a maximum plane tree in a multipartite geometric graph. It is not known whether or not this problem is NP-hard. A natural open problem is to improve any of the presented approximation ratios. Specifically, when the number of sets in the partition is more than two, we conjecture that the length of the longest star is at least 1/3 times the length of a maximum spanning tree.

We also presented exact algorithms for some special bichromatic input point sets. Providing an $o(n^2)$ algorithm for the two red point case is open.

References

[1] N. Alon, S. Rajagopalan, and S. Suri. Long non-crossing configurations in the plane. *Fundamenta Informaticae*, 22(4):385–394, 1995. Also in *Proceedings of the 9th ACM Symposium on Computational Geometry* (SoCG), 257–263, 1993.

[2] A. Biniaz, P. Bose, D. Eppstein, A. Maheshwari, P. Morin, and M. Smid. Spanning trees in multipartite geometric graphs. *CoRR*, abs/1611.01661, 2016. Also submitted to *Algorithmica.*

[3] M. G. Borgelt, M. J. van Kreveld, M. Löffler, J. Luo, D. Merrick, R. I. Silveira, and M. Vahedi. Planar bichromatic minimum spanning trees. *Journal of Discrete Algorithms*, 7(4):469–478, 2009.

[4] A. Dumitrescu and C. D. Tóth. Long non-crossing configurations in the plane. *Discrete & Computational Geometry*, 44(4):727–752, 2010. Also in *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science* (STACS), 311–322, 2010.

[5] C. L. Monma, M. Paterson, S. Suri, and F. F. Yao. Computing Euclidean maximum spanning trees. *Algorithmica*, 5(3):407–419, 1990.

A Proof of $f(x,\alpha) \geqslant 0$

We want to show that

$$f(x,\alpha) = \sqrt{1+x^2-2x\cos(\pi/3-\alpha)} + \sqrt{1+x^2-2x\cos\alpha} - x \geqslant 0$$

for all $\sqrt{3}/2 \leqslant x \leqslant \sqrt{3}$ and $0 \leqslant \alpha \leqslant \frac{\pi}{6}$. From elementary trigonometry, we have

$$\cos\left(\frac{\pi}{3}-\alpha\right) = \frac{1}{2}\cos\alpha + \frac{\sqrt{3}}{2}\sin\alpha,$$

from which $f(x,\alpha)$ can be re-written as

$$f(x,\alpha) = \sqrt{1+x^2-x\cos\alpha-\sqrt{3}\,x\sin\alpha} + \sqrt{1+x^2-2x\cos\alpha} - x.$$

Let us solve the equation $f(x,\alpha)=0$, which corresponds to

$$\sqrt{1+x^2-x\cos\alpha-\sqrt{3}\,x\sin\alpha} + \sqrt{1+x^2-2x\cos\alpha} = x.$$

By squaring on both sides, we find

$$2+2x^2-3x\cos\alpha-\sqrt{3}\,x\sin\alpha+2\sqrt{1+x^2-x\cos\alpha-\sqrt{3}\,x\sin\alpha}\sqrt{1+x^2-2x\cos\alpha} = x^2,$$

which we write as

$$2\sqrt{1+x^2-x\cos\alpha-\sqrt{3}\,x\sin\alpha}\sqrt{1+x^2-2x\cos\alpha} = -2-x^2+3x\cos\alpha+\sqrt{3}\,x\sin\alpha.$$

Squaring once more, we find

$$4\left(1+x^2-x\cos\alpha-\sqrt{3}\,x\sin\alpha\right)\left(1+x^2-2x\cos\alpha\right) = \left(-2-x^2+3x\cos\alpha+\sqrt{3}\,x\sin\alpha\right)^2,$$

which is equivalent to

$$4\left(1+x^2-x\cos\alpha-\sqrt{3}\,x\sin\alpha\right)\left(1+x^2-2x\cos\alpha\right) - \left(-2-x^2+3x\cos\alpha+\sqrt{3}\,x\sin\alpha\right)^2 = 0,$$

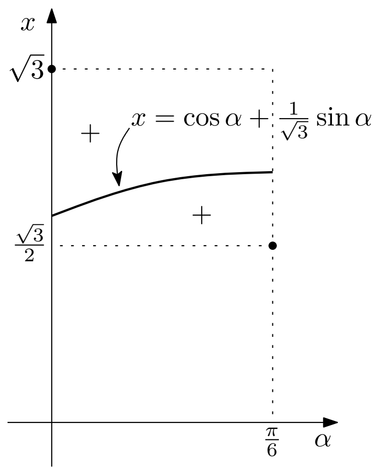
which can be factored into

$$3x^2\left(x-\left(\cos\alpha+\frac{1}{\sqrt{3}}\sin\alpha\right)\right)^2 = 0.$$

Since $x>0$ and

$$f\left(\cos\alpha+\frac{1}{\sqrt{3}}\sin\alpha,\alpha\right) = 0,$$

we have that $f(x,\alpha)=0$ if and only if $x=\cos\alpha+\frac{1}{\sqrt{3}}\sin\alpha$. Therefore, on its domain, f is equal to 0 precisely on the curve $x=\cos\alpha+\frac{1}{\sqrt{3}}\sin\alpha$ and nowhere else. Thus, below this curve, f is everywhere positive or everywhere negative. Since $f\left(\frac{\sqrt{3}}{2},\frac{\pi}{6}\right) = 1-\frac{\sqrt{3}}{2} > 0$, f is everywhere positive below the curve. Similarly, since $f(\sqrt{3},0) = \sqrt{4-\sqrt{3}}-1 > 0$, f is everywhere positive above the curve. Therefore, $f(x,\alpha)\geqslant 0$ for all $\sqrt{3}/2 \leqslant x \leqslant \sqrt{3}$ and $0 \leqslant \alpha \leqslant \frac{\pi}{6}$.



x
$\sqrt{3}$
$x=\cos\alpha+\frac{1}{\sqrt{3}}\sin\alpha$
$+$
$+$
$\frac{\sqrt{3}}{2}$
$\frac{\pi}{6}$
α