Planar Bichromatic Minimum Spanning Trees*

Magdalene G. Borgelt† Marc van Kreveld† Maarten Löffler† Jun Luo† Damian Merrick‡
Rodrigo I. Silveira† Mostafa Vahedi†

Abstract

Given a set S of n red and blue points in the plane, a *planar bichromatic minimum spanning tree* is the shortest possible spanning tree of S, such that every edge connects a red and a blue point, and no two edges intersect. Computing this tree is NP-hard in general. We present an $O(n^3)$ time algorithm for the special case when all points are in convex position. For the general case, we present a factor $O(\sqrt{n})$ approximation algorithm.

1 Introduction

Let S be a set of n points in the plane, where every point has one of two possible colors (red or blue). In computational geometry, several papers have discussed problems that concern such a bichromatic input, see for instance Kaneko and Kano [5] for an overview. This paper discusses bichromatic spanning trees. We obtain a spanning tree T of S by finding a set of $n-1$ edges, which connect points of different colors ("color conforming") and form an acyclic connected component. If T does not contain intersections it is a *planar* spanning tree. In this paper we assume that no three points are collinear, otherwise a planar bichromatic spanning tree does not always exist.

A *minimum weight spanning tree* (MST) of S is a spanning tree of minimum total length. Note that a MST needs not be unique. It is well known that the (monochromatic) MST of a set of points in the plane can be found using a greedy algorithm like Kruskal's [6]. Kruskal's algorithm adds edges in the order of increasing length, and discards edges that would create a cycle in the graph built so far. The (monochromatic) MST of a set of points or line segments in the plane cannot contain intersections [2, 4]. It is shown that the problem of finding the Euclidean (monochromatic) MST of a set of n points in d-dimensional space is related to the problem of finding the bichromatic closest pair among n red and m blue points in d-dimensional space [1].

Recently it was shown that a color conforming spanning tree of S can always be found [4]. It was also shown by illustration that the MST of a given point set S may contain intersections if one uses a greedy algorithm like Kruskal's [3]. Modifying Kruskal's algorithm to check for intersections and discarding an edge if it causes an intersection, leads to a greedy algorithm which we will refer to as the *greedy planar algorithm* or the *augmented Kruskal algorithm*. It was shown that the greedy planar algorithm does not always yield the optimal planar solution [3], and can even be a linear factor off. The problem of finding a superlinear bound for the ratio of the weight of the greedy planar solution to the weight of the optimal planar solution was left open. Another open problem was to find an approximation algorithm for the planar bichromatic MST of S.

In this paper we show that a planar bichromatic spanning tree of a set of red and blue points may not always be obtainable using a greedy algorithm like the augmented Kruskal algorithm (hence, this algorithm has no approximation factor at all). We can show that the planar bichromatic minimum spanning tree problem is NP-hard in the general case (the proof is in the full paper), but we show that the optimal tree can be constructed in cubic time when the points are in convex position. Finally, we present an approximation algorithm that computes an $O(\sqrt{n})$-approximation in $O(n^2)$ time.

2 Greedy Planar Bichromatic Spanning Trees

One approach to create planar bichromatic spanning trees is by using a greedy algorithm. The algorithm proposed by Kruskal [6] and augmented for bichromatic trees [3] is an example of this. In this section we show that a greedy algorithm may in some cases not find any planar bichromatic spanning tree at all, because at some stage there are points that cannot be connected to any point of the other color anymore.

*This research has been partially funded by the Netherlands Organisation for Scientific Research (NWO) under FOCUS/BRICKS grant number 642.065.503, and also through the project GOGO.

†Department of Information and Computing Science, Utrecht University, the Netherlands, { magdalene, marc, loffler, ljroger, rodrigo, vahedi}@cs.uu.nl

‡National ICT Australia; School of Information Technologies, University of Sydney, Australia, dmerrick@it.usyd.edu.au

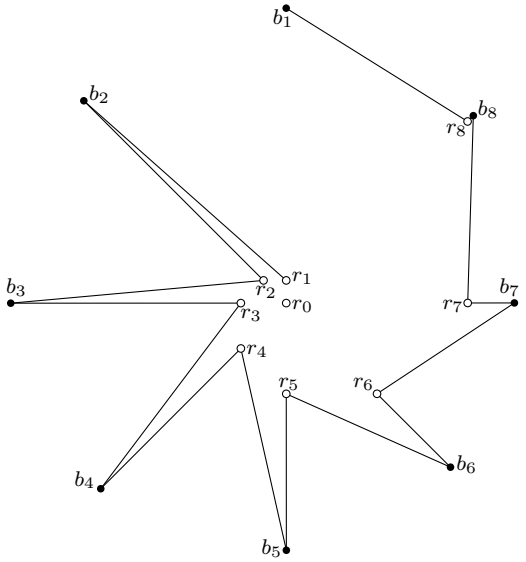

Figure 1: An example of a set of bichromatic vertices and bichromatic edges that are selected by an augmented Kruskal algorithm.

Specifically, we show that Kruskal's algorithm may get stuck in this way.

Consider the set of nine red points (r_i, $i = 0, \ldots, 8$) and eight blue points (b_i, $i = 1, \ldots, 8$), as in Figure 1. Here the point b_i lies slightly to the left of the directed line from r_0 to r_i, for any i, so once all the black edges in the figure have been included by some greedy algorithm, the central point r_0 cannot be connected to any blue point anymore, and therefore the existing tree cannot be extended to a valid planar bichromatic spanning tree.

The augmented Kruskal algorithm adds edges in the order of increasing length, discarding edges that would cause intersections as well as edges that would create a cycle in the graph built so far. In this situation, the algorithm will create the edges shown in the figure. This is because for any edge from r_0 to one of the blue points, there is another edge crossing it that is shorter and therefore will be chosen first.

To see why this happens, consider for any i the group of points r_{i-1}, r_i, b_{i-1}, and b_i. Of the two edges (r_{i-1}, b_i) and (b_{i-1}, r_i) the former must be shorter in order to obtain the desired structure (that is, a structure in which the edge (r_{i-1}, b_i) shields r_0 from seeing b_{i-1}, so that (r_0, b_{i-1}) would lead to an intersection). To simplify the situation, let us assume that r_0, r_{i-1}, and b_{i-1} are collinear, and that r_0, r_i, and b_i are collinear. In addition, let us assume that the angle between the two directions is exactly $\frac{\pi}{4}$. If we place the red points on a spiral where the distance between r_0 and r_i is $\sqrt{2}$ times larger than the distance between r_0 and r_{i-1}, and we place the b_i at carefully chosen locations at the outside of the construction, we get the required property.

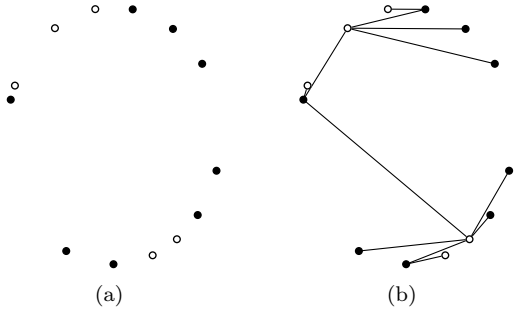

Figure 2: Example of the input (left) and output (right) for a set of bichromatic points in convex position.

Theorem 1 *A planar bichromatic minimum spanning tree of a set of red and blue points may not always be obtainable using a greedy algorithm like the augmented Kruskal algorithm.*

3 NP-hardness

We show in the full paper that the problem is NP-hard, by reduction from planar 3-SAT.

Theorem 2 *The problem of computing a planar bichromatic minimum spanning tree of a set of red and blue points, is NP-hard.*

4 Dynamic programming for points in convex position

If we have a set of red and blue points in the plane that are in convex position, then we can compute the planar bichromatic minimum spanning tree in $O(n^3)$ time, see Figure 2(b). This is because in this case, any edge of the tree partitions the plane into two independent subproblems.

We are given a set of points in convex position, see Figure 2(a). Let them be ordered cyclicly counterclockwise. For any pair of points p and q (possibly of the same color, but not necessarily) we define T_{pq} as the planar bichromatic minimum spanning tree of the points $\{p, \ldots, q\}$ (the set of points encountered when walking from p to q, in counterclockwise order). Then the answer for the whole point set is T_{pq}, for any pair of consecutive points q and p.

We can compute all lengths of T_{pq} by dynamic programming, starting from the simplest ones. The length of T_{pp} is zero. For a given pair of points $p \neq q$ we observe the following cases:

- p and q are neighbors. If they have different colors, then $T_{pq} = \overline{pq}$. Otherwise, T_{pq} does not exist.
- p and q are of the same color. There exists a point r between p and q such that $T_{pq} = T_{pr} \cup T_{rq}$, or T_{pq} does not exist.
- p and q are of different colors, but they are not connected by a direct edge. There exists a point r between p and q such that $T_{pq} = T_{pr} \cup T_{rq}$.
- p and q are of different colors, and they are connected by a direct edge. There exist two neighboring points r and s (one of which may be p or q), such that $T_{pq} = \overline{pq} \cup T_{pr} \cup T_{sq}$.

In any case, we can compute T_{pq} in linear time by guessing the position of r and taking the best over the possible results. Since there are $O(n^2)$ possible choices for p and q, we solve the problem in $O(n^3)$ time.

Theorem 3 *A planar bichromatic minimum spanning tree of a set of n red and blue points in convex position can be computed in $O(n^3)$ time.*

5 An $O(\sqrt{n})$-approximation

Since the problem cannot be solved exactly in polynomial time unless P = NP, we will now describe an approximation algorithm that computes a planar bichromatic spanning tree that is at most $O(\sqrt{n})$ times larger than the optimal one, in polynomial time.

We start by taking a bounding square around the point set, such that the set either touches the square on both the top and bottom edges, or on both the left and right edges. After this we scale the input, such that the bounding square becomes the unit square.

Lemma 4 *The length of the optimal planar bichromatic spanning tree of the scaled point set is at least 1.*

Proof. There are two points on opposite borders of the bounding squares, so they are at least 1 away from each other. □

Next, we create a grid by dividing the unit square into $\sqrt{n} \times \sqrt{n}$ square cells, of side $\frac{1}{\sqrt{n}}$. The point set is also divided by the grid, see Figure 3(b).

Lemma 5 *If m of the n grid cells contain at least a point, then the length of the optimal planar bichromatic spanning tree is at least $O(\frac{m}{\sqrt{n}})$.*

Proof. If m grid cells contain a point, then at least $\frac{1}{4}m$ cells that are not adjacent contain a point. To connect these points with any spanning tree, a connection of length at least $\frac{1}{\sqrt{n}}$ is needed per grid cell. □

We classify the cells according to what points are inside them. There are three possibilities. Each grid cell is either empty, contains only points of one color, or contains points of both colors.

We will now define a set of *core regions* as follows. A core region is either a single grid cell, two adjacent grid cells or four grid cells adjacent to a grid vertex. Each core region contains both red and blue points. Apart from that, the following crucial property should hold.

Crucial Property: Every point is either a distance of at most $O(1/\sqrt{n})$ (a constant number of grid cells) away from a core region, or a distance of at least $1/\sqrt{n}$ (one grid cell) away from the closest point of the other color.

This crucial property guarantees the approximation factor: points in the first category will be connected by an edge of $O(1/\sqrt{n})$ length, so all of these together will not be more than $O(\sqrt{n})$. Points in the second category will be connected by an edge of length at most $O(1)$, but in the optimal solution they are connected by some edge of at least length $\Omega(1/\sqrt{n})$, so they are at most a factor $O(\sqrt{n})$ too long.

We compute the core regions iteratively as follows.

- Every grid cell that contains points of two colors is a core region.
- For every grid cell that contains only points of one color and that is not adjacent to a core region, do the following: if it has a neighboring grid cell that contains points of the other color, make a new core region out of those two grid cells (and possibly two more if they were vertex-adjacent), otherwise do nothing.

This procedure results in a set of core regions with the properties just described. Figure 3(c) gives an example.

We want to get rid of the core regions that touch each other, so we compute a maximal independent set of the core regions with respect to the adjacency (edge or vertex) relation. After this we have a smaller set of core regions, with the property that the regions are separated by a band at least one grid cell wide. Furthermore, the discarded core regions are all adjacent to a core region that belongs to the independent set, so the crucial property still holds.

Next, we compute the Voronoi diagram of the centers of the core regions. This is a subdivision of the plane

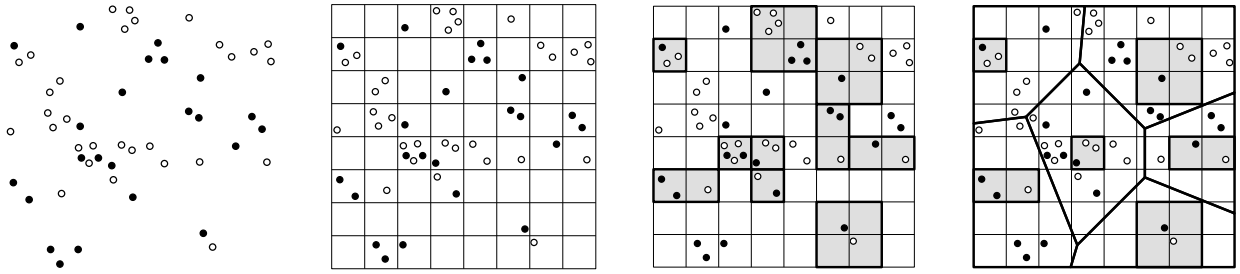

Figure 3: (a) A set of red and blue points. (a) The points divided by a grid. (b) Core regions are shaded and bounded by fat edges. (c) The Voronoi regions of an independent subset of the core regions.

into convex cells that all have one core region inside them, because any point inside a core region is at most $\sqrt{2}/\sqrt{n}$ away from the center of its own region, and at least $1.5/\sqrt{n}$ away from the center of any other region, see Figure 3(d).

Inside each Voronoi cell we compute a factor $O(\sqrt{n})$ approximation spanning tree as follows. First we compute any tree for the points inside the core region. Then we sort the other points on their distance to the core region, and add them in that order. This is always possible, as the following lemma shows.

Lemma 6 *If T is a planar bichromatic spanning tree, and q is a point in one of the colors that is outside the convex hull of T, then T can be extended by one more edge to a planar bichromatic spanning tree that includes q.*

The insertion order ensures that the length of each new connection is at most the distance to the core region plus the size of the core region times some constant. This means that the points that are near the borders of the core regions will have a connection of length $O(\frac{1}{\sqrt{n}})$, which is what we need for the approximation.

After building the trees inside all Voronoi cells, we combine them and extend them to one tree using the $O(n \log n)$ algorithm described in [4]. The length of the edges used by the algorithm to connect each component does not matter, because the number of Voronoi cells is at most the number of grid cells with points inside them, m, and even if all these connections are as bad as possible, we still only have a length of at most m. By Lemma 5, this is a factor $O(\sqrt{n})$ approximation of the optimum.

All steps of the algorithm take quadratic time in the number of occupied grid cells, so the algorithm runs in $O(n^2)$ time.

Theorem 7 *An $O(\sqrt{n})$-approximation of a planar bichromatic minimum spanning tree of a set of n red and blue points can be computed in $O(n^2)$ time.*

6 Conclusions

We studied the problem of computing a planar bichromatic minimum spanning tree of a set of red and blue points in the plane. We showed that this problem is NP-hard, and gave an $O(\sqrt{n})$-approximation algorithm. An interesting open problem is whether a constant factor approximation algorithm for this problem that runs in polynomial time exists.

References

[1] P. K. Agarwal, H. Edelsbrunner, O. Schwarzkopf, and E. Welzl. Euclidean minimum spanning trees and bichromatic closest pairs. In *Discrete and Computational Geometry*, 6(5):407-422, 1991.

[2] P. Bose and G. Toussaint. Growing a Tree from its Branches. *Journal of Algorithms* 19(1):86-103, 1995.

[3] M. Grantson, H. Meijer, and D. Rappaport. Bi-Chromatic Minimum Spanning Trees. In *Proc. 21st European Workshop on Computational Geometry (EuroCG05)*, pages 199–202, 2005.

[4] F. Hurtado, M. Kano, D. Rappaport and C.D. Tòth. Encompassing Colored Crossing-Free Geometric Graphs. In *16th Canadian Conference on Computational Geometry*, pages 48–52, 2004.

[5] A. Kaneko and M. Kano. Discrete Geometry on Red and Blue Points in the Plane – A Survey. In *Discrete and Computational Geometry* (B. Aronov et al., eds.), Springer-Verlag, Berlin, pages 551–570, 2004.

[6] J.B. Kruskal. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. In *Proc. of the American Mathematical Society,* 7:48–50, 1956.