Planar Bichromatic Minimum Spanning Trees[*]

Magdalene G. Borgelt[1] Marc van Kreveld[2] Maarten Löffler[2]
Jun Luo[2] Damian Merrick[3] Rodrigo I. Silveira[2] Mostafa Vahedi[2]

[1] European Centre for Soft Computing
Mieres, Asturias, Spain
magdalene@borgelt.net

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

[3] National ICT Australia, School of Information Technologies,
University of Sydney, Australia
dmerrick@it.usyd.edu.au

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. We show that computing this tree is NP-hard in general. For points in convex position, a cubic-time algorithm can be easily designed using dynamic programming. We adapt such an algorithm for the special case where the number of red points (m) is much smaller than the number of blue points (n), resulting in an $O(nm^2)$ time algorithm. For the general case, we present a factor $O(\sqrt{n})$ approximation algorithm that runs in $O(n \log n \log \log n)$ time. Finally, we show that if the number of points in one color is bounded by a constant, the optimal tree can be computed in polynomial time.

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, like red-blue intersection ([2, 19] and many more), red-blue separation (e.g. [4, 8, 10, 11, 15]), and red-blue connection problems (e.g. [1, 3, 5, 14]). See Kaneko and Kano [16] for an overview. This paper discusses red-blue, or bichromatic, spanning trees. We obtain a spanning tree T of S by finding a set of $n-1$ edges that connect pairs of points of different colors ("color conforming" or "bichromatic") 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 an MST needs not be unique. It is well known that the (monochromatic) MST of a set of

[*]Partially funded by the Netherlands Organization for Scientific Research (NWO) under BRICKS/FOCUS grant number 642.065.503, and under the Open Competition project GOGO.

points in the plane can be found using a greedy algorithm like Kruskal's [17]. 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 [9, 14].

Regarding the bichromatic version, it has been shown that a set of non-intersecting, bichromatic line segments can always be extended to a planar bichromatic spanning tree [13, 14]. If loose points occur as well, then no planar bichromatic spanning tree may be possible. It has also been shown that the bichromatic MST of a given point set S may contain intersections if one uses a greedy algorithm like Kruskal's [12]. 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*; it does not always yield the optimal planar solution [12], 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. Actually, in some cases a greedy algorithm may not find any planar bichromatic spanning tree at all, because at some stage there might be points that cannot be connected to any point of the other color anymore. An example of a point set that results in this situation is shown in Figure 1. After several steps of the greedy planar algorithm, the red point r_0 cannot be connected to any blue point anymore. More details on the construction of such a point set are given in [7]. Another problem left open in [12] is finding an approximation algorithm for the planar bichromatic MST of S.

In this paper we show that the planar bichromatic minimum spanning tree problem is NP-hard in the general case. However, for points in convex position it can be computed in cubic time using dynamic programming. We adapt this approach, for points in convex position, to the special case where the number of red points (m) is significantly less than the number of blue points (n), resulting in an $O(nm^2)$ time algorithm. Then we present an approximation algorithm that computes an $O(\sqrt{n})$-approximation in $O(n \log n \log \log n)$ time. Finally, we discuss computing optimal planar bichromatic minimum spanning trees with only few red points and n blue points. For two red points and $n-2$ blue points we give an $O(n \log n)$ time algorithm, and for $k > 2$ red

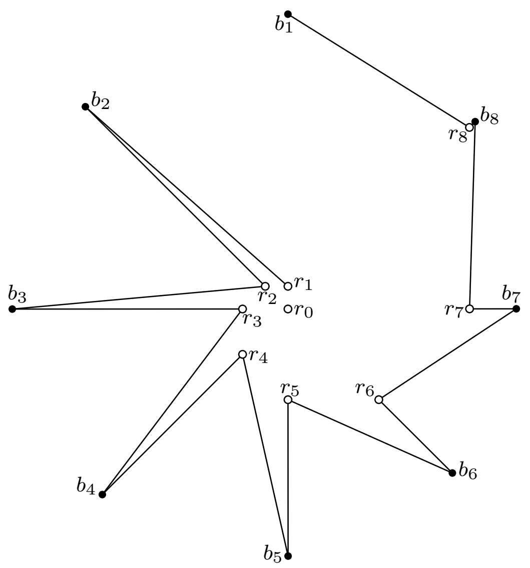

Figure 1: An example set of bichromatic vertices and the bichromatic edges that are selected by the greedy planar algorithm, an augmented version of Kruskal's algorithm. Once the algorithm reaches this situation it is not possible to continue because r_0 cannot be connected to any blue point.

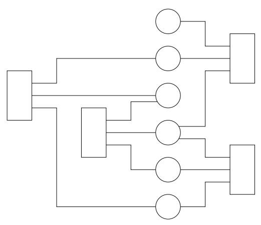

Figure 2: An instance of planar 3-SAT where variables are shown as circles and clauses as rectangles.

points and $n-k$ blue points, for a constant k, we give an $n^{O(k^5)}$ time algorithm.

2 Computing the planar bichromatic minimum spanning tree is NP-hard

We prove that the general planar bichromatic minimum spanning tree problem is NP-hard by reduction from planar 3-SAT [18]. Planar 3-SAT is an NP-complete variant of 3-SAT, where there is the additional constraint that the dependency graph is planar. This graph has the variables and clauses of the formula as nodes, and there is an edge between a variable node and a clause node if the variable occurs in the clause (see Figure 2).

We base our construction on pairs of one red and one blue point that are very close together. We call such a pair a site. We place a lot of these sites in the plane, together with a number of single red points. The shortest bichromatic spanning tree that we could hope for in this situation is the classical minimum spanning tree of this set of sites (plus some small overhead, which we can make as small as we want). However, this may not be realizable as a planar spanning tree, because connections between pairs can interfere with each other. The idea is that in our construction, this is possible exactly when a given 3-SAT formula can be satisfied. We will also include pairs of sites at equal distance, so the classical MST will not be uniquely defined, but of course its value will.

Given a planar embedding of a 3-SAT instance (see Figure 2), we will construct gadgets to represent the different variables and clauses of this instance.

The main gadget we need in the construction is the variable chain, see Figure 3(a). This gadget consists of a "chain" of sites, where the distance between all consecutive sites is equal and the distances between any other two sites is larger. Now, between every two consecutive sites, there are two possible edges to connect them: we either connect the blue point of the first site with the red point of the second site, or the other way around. The chain is constructed in such a way

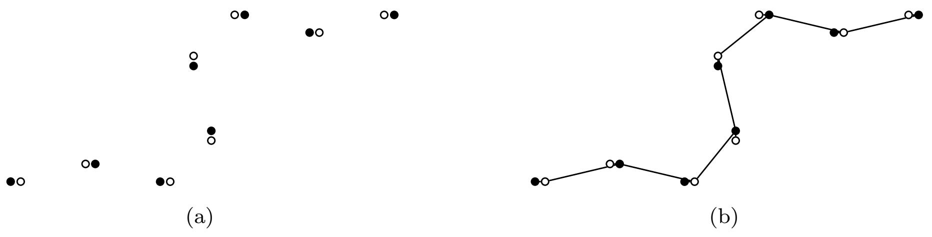

Figure 3: (a) Input points of a variable chain. (b) If the red (open) point of the leftmost site is chosen for the connection to the next site, then there is only one way to connect the whole chain.

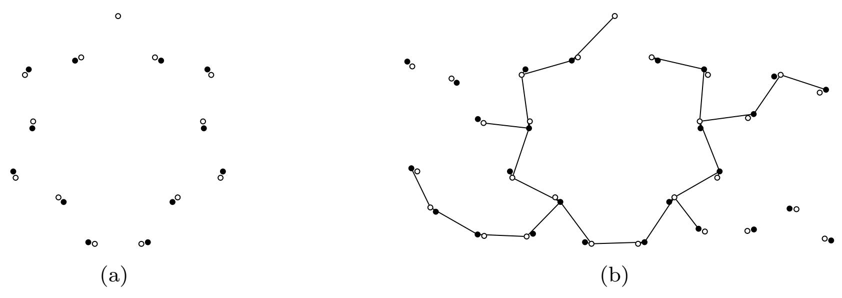

Figure 4: (a) The input for a variable. (b) One of the two possible states of the variable, say `true`, propagates through some of the connected chains.

that if we choose a certain connection at a given point, this forces the choice of connections in one direction along the chain, see Figure 3(b).

The next thing we need are variable gadgets. We represent variables by a circular chain, with one single red point and red/blue pairs elsewhere, see Figure 4(a). Here the distances between all consecutive sites are equal, but the distance from the single red point to its neighboring sites is larger (e.g., twice as large). This means that, in an optimal solution, we need all connections between consecutive sites and exactly one of the two connections to the red point. If we choose any of the two possibilities, this choice propagates through the chain and the whole tree is fixed, see Figure 4(b). This means that there are only two possible ways to connect this gadget, which will correspond to the `true` and `false` states of a variable in the 3-SAT instance. Now we can 'tap off' the state of this variable towards any clause where we need it by connecting chains to the inner sites (of course the variable gadget can contain more sites if we need the variable more often), see Figure 4(b). Depending on where and how we connect the chains to the variable, it will propagate only when the variable is in the `true` state or only when the variable is in the `false` state.

Then we need a clause gadget. We make these by taking a single red point, and three red/blue pairs at a fairly large distance from it, with the red point of the pair towards the central red point (see Figure 5(a)). We connect these sites with chains to their respective variables. In an optimal spanning tree, the central red point will be connected to exactly one of the three sites. However, it cannot connect to a site if the respective chain is in the wrong state. If the variable gadget is set to one state, the forced edges are propagated through some of the chains, and when it is set to the other state, it is propagated through the other chains. So, we just need to propagate the `true` value to the clauses where the variable is used as `false`, and vice versa.

Finally we need to connect all variables to each other by some fixed part of the tree, because the whole construction needs to be a tree and not a forest. The variables of any planar 3-SAT instance

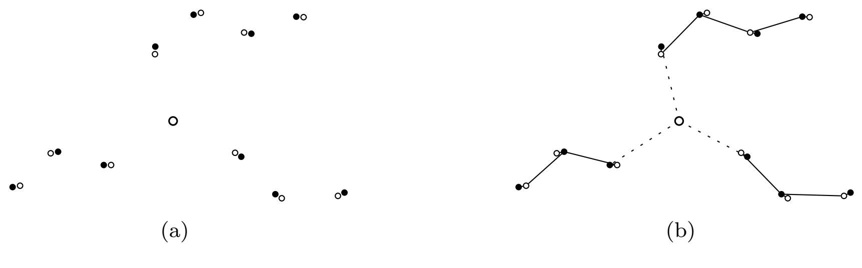

Figure 5: (a) Input points of a clause gadget. (b) If all three chains have their fixed edges propagated towards the clause, the red center point cannot be connected anymore using the desired distance.

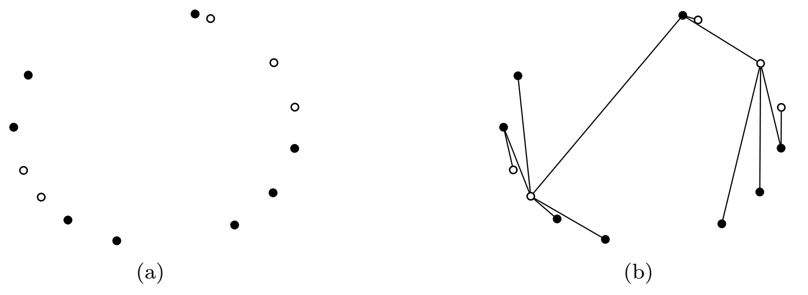

Figure 6: Example of the input (a) and output (b) for a set of bichromatic points in convex position. Any edge of a minimum spanning tree partitions the plane into two independent subproblems.

can always be laid out as in Figure 2, and by the design of variable gadgets, these connections can easily be made. Also, we need to make sure that the distance between different parts of the construction is large enough to avoid shortcuts.

Now we have a situation where a planar bichromatic spanning tree of the same length (plus some arbitrarily small overhead) as the classical, monochromatic minimum spanning tree of the set of sites and single red points exists, if and only if the 3-SAT instance is satisfiable.

Theorem 1 *The problem of computing a planar bichromatic minimum spanning tree of a set of red and blue points is NP-hard.*

3 Dynamic programming for points in convex position

If we have a set of n 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. This is because in this case, any edge of the tree partitions the plane into two independent subproblems, see Figure 6. The following result can be obtained by a straightforward application of dynamic programming (details can be found in [7]).

Theorem 2 *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.*

When the number of red points (m) is much smaller than the number of blue points (n), we can use a more involved dynamic programming algorithm in order to reduce the running time to $O(nm^2)$. The details are explained next.

Let S be a point set of points in convex position with m red points and n blue points, where m is considerably smaller than n, that is, $m \ll n$. The points are assumed to be ordered cyclicly counterclockwise. Let $T_{i,j}$ be the planar bichromatic minimum spanning tree of the points $\{p_i, \dots, p_j\}$ (in counterclockwise order). We will only compute $T_{i,j}$ if at least one of p_i, p_j is red. Furthermore, as an auxiliary problem we will define $R_{i,j}$ as a pair of disjoint, planar bichromatic spanning trees of the points $\{p_i, \dots, p_j\}$ of minimum total length, such that p_i and p_j are in different trees. We will only compute $R_{i,j}$ if both p_i and p_j are red. Note that the table used to compute T has $O(nm)$ entries and the table for R has $O(m^2)$ entries.

Notice that for $R_{i,j}$, disjointness implies that there is a gap between two consecutive points p_k and p_{k+1} in $p_i, \dots, p_j$ so that $R_{i,j}$ is the union of $T_{i,k}$ and $T_{k+1,j}$. So if all trees T within $p_i, \dots, p_j$ are known, then $R_{i,j}$ can be determined in $O(n)$ time by trying all $j - i$ possibilities of the gap.

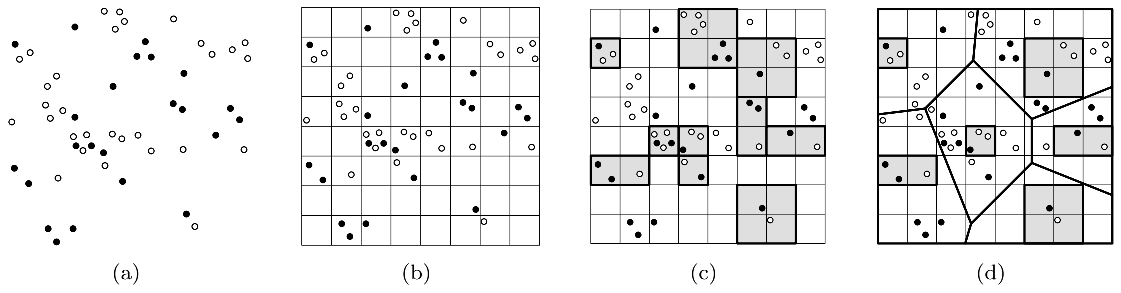

Figure 7: (a) A set of red and blue points. (b) The points divided by a grid. (c) Core regions are shaded and bounded by fat edges. (d) The Voronoi regions of a maximal independent subset of the core regions.

To compute the entries of the table for T, we observe the following cases for a given pair of points p_i and p_j, at least one of which is red. The base cases are all trees $T_{i,j}$ where p_i is red and $p_{i+1}, \ldots, p_j$ are blue, or p_j is red and $p_i, \ldots, p_{j-1}$ are blue. These cases can easily be solved in $O(n)$ time in total. The general cases are the following.

- p_i and p_j are both red. Then either there exists a blue point p_b between p_i and p_j such that $T_{i,j} = T_{i,b} \cup T_{b,j}$, or $T_{i,j}$ does not exist. We try all the possible blue points in $O(n)$ time. Notice that this is done only for the red-red combinations, and there are only $O(m^2)$ of them in total.
- p_i is red and p_j is blue, or vice versa, and they are not directly connected. Then there exists a red point p_r between p_i and p_j such that $T_{i,j} = T_{i,r} \cup T_{r,j}$. We try all the possible red points in $O(m)$ time.
- p_i is red and p_j is blue, and they are directly connected. We consider two possibilities. (1) p_j is not connected to any other red point than p_i in $T_{i,j}$. Then $T_{i,j} = T_{i,j-1} \cup \overline{p_i p_j}$. (2) p_j is connected to at least one red point in $p_{i+1}, \ldots, p_{j-1}$. Then there is such a red point p_r with the lowest index, and $T_{i,j} = T_{r,j} \cup \overline{p_i p_j} \cup R_{i,r}$. The first possibility is a single option, and the second possibility gives $O(m)$ choices for r.
- p_i is blue and p_j is red, and they are directly connected. This case is completely analogous to the previous case.

We observe that none of the subproblems requires a solution to $T_{i,j}$ where both p_i and p_j are blue. The final answer can be found in $T_{i+1,i}$, where at least one of p_i, p_{i+1} is red. The table for T has m rows and $n + m$ columns, each red-blue entry is filled in $O(m)$ time, and each red-red entry is filled in $O(n)$ time. The table for R has m rows and m columns, and each entry is filled in $O(n)$ time. Therefore the total running time is $O(nm^2)$.

Theorem 3 *A planar bichromatic minimum spanning tree of a set of m red and n blue points in convex position, given in cyclic order, can be computed in $O(nm^2)$ time.*

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

In this section we present 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 tight axis-parallel bounding square around the input point set, and scale the square and the input so that the bounding square has unit size.

Lemma 1 *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. $\boxtimes$

Next, we create a grid by dividing the unit square into $\sqrt{n} \times \sqrt{n}$ square cells, of side $1/\sqrt{n}$. The point set is also divided by the grid, see Figure 7(b).

Lemma 2 *If m of the n grid cells contain at least a point, then the length of the (optimal) planar bichromatic minimum spanning tree is at least $\Omega(m/\sqrt{n})$.*

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

We classify the cells according to what points are inside them. There are three possibilities. Each grid cell is either empty, or 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, or 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 two properties should hold for a set of core regions.

Distance 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 in the set, or a distance of at least $1/\sqrt{n}$ (one grid cell) away from the closest point of the other color.

Separation Property: Every two core regions in the set are at least $1/\sqrt{n}$ apart.

The Distance Property guarantees the approximation factor: points in the first category will be connected by an edge of $O(1/\sqrt{n})$ length, so the sum of these lengths 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 start by computing a set of non-overlapping candidate core regions that has the Distance Property, but does not necessarily have the Separation Property. From this set we select a subset that satisfies the Separation Property, while not violating the Distance Property. We compute a set of candidate 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 an adjacent 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 candidate core regions that are non-overlapping and have the Distance Property. Note that the set of candidates is not uniquely defined. Figure 7(c) gives an example.

To satisfy the Separation Property, 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 Distance Property still holds.

Next, we compute the Voronoi diagram of the centers of the core regions. This is a subdivision of the plane 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 7(d).

Inside each Voronoi cell we compute a factor $O(\sqrt{n})$ approximation planar bichromatic spanning tree as follows. If we pick any point p inside a core region, we can list all other points $q_1, q_2, ..., q_k$ in the Voronoi cell of this core region according to their distances to p. We add those points in the listed order. Let BST_i be the bichromatic spanning tree constructed after q_i is added (note that BST_i is not a tree if $p, q_1, q_2, ..., q_i$ are all the same color). When q_{i+1} is added, suppose we can always find a point q_j among $p, q_1, q_2, ..., q_i$ such that the edge $\overline{q_{i+1}q_j}$ added to BST_i forms a new bichromatic spanning tree BST_{i+1}. Then we have following lemma:

Lemma 3 *BST_k is $O(\sqrt{n})$ factor approximation of the optimal solution for $\{p, q_1, \ldots, q_k\}$.*

Proof: When we add q_{i+1}, consider a circle with center p and radius $|\overline{q_{i+1}p}|$. All points of $p, q_1, q_2, ..., q_i$ are inside this circle, one of which is q_j. Therefore $|\overline{q_{i+1}q_j}| \leq 2|\overline{q_{i+1}p}|$. If q_{i+1} is a point of the first category in Distance Property, then $|q_{i+1}p| = O(1/\sqrt{n})$ which means $|q_{i+1}q_j| = O(1/\sqrt{n})$. If q_{i+1} is a point of the second category in Distance Property, then $|q_{i+1}q_j| = O(1)$. But in the optimal solution q_{i+1} is connected by some edge of at least length $\Omega(1/\sqrt{n})$, so it is at most a factor $O(\sqrt{n})$ too long. ⊠

Next we show that we can always construct BST_{i+1} from BST_i by adding a bichromatic edge. Suppose p is red and q_h ($1 \leq h \leq k$) is the first blue point in the list. Then $BST_h = \overline{pq_h} \cup \overline{q_1q_h} \cup \ldots \cup \overline{q_{h-1}q_h}$ which is a star-shaped tree. We divide the Voronoi cell into convex cells as follows: for the first added edge $\overline{pq_h}$, we cut it with the line through p and q_h. For all other edges $\overline{q_iq_h}$ with $2 \leq i \leq h-1$, we cut with a ray from q_h in the direction of q_i until it hits the boundary of the Voronoi cell. This way, the Voronoi cell is partitioned in $h+1$ convex cells with the property that each cell has at least one red point and one blue point on its boundary (see Figure 8(a)). Then we add the remaining points $q_{h+1}, \ldots, q_k$ one by one. We will maintain a convex partition CP_i with the properties that all spanning tree edges of BST_i are in the boundaries of cells, and each cell has a red and a blue point from $\{p, q_1, \ldots, q_i\}$ on the boundary. We maintain CP_i in a point location structure that allows insertions of vertices and edges [6].

Suppose that we have added all points up to q_i, and we wish to compute BST_{i+1} from BST_i. To add q_{i+1}, perform a planar point location query to determine the cell it is in. Take the point q_j with different color on its boundary, and add the edge $\overline{q_{i+1}q_j}$ to BST_{i+1}. Since the cell is convex, no crossings can be created. Now we need to compute the convex partition CP_{i+1} from CP_i and update the point location structure. The cell of CP_i containing $\overline{q_{i+1}q_j}$ must be split into two by the line through q_{i+1} and q_j. One of the intersection points is q_j, the other we find by a tandem walk along the boundary of the cell, starting at q_j and in both directions. Let this point be t. We add t as a vertex to CP_i, and add the edges $\overline{q_{i+1}q_j}$ and $\overline{q_{i+1}t}$ as well, resulting in CP_{i+1}. Since q_{i+1} and q_j are on the boundaries of both new cells, they both have a red point and a blue point on their boundaries.

The dynamic planar point location structure of Baumgarten et al. [6] supports point location and insertions in $O(\log n \log\log n)$ time. For the tandem walk, we alternatingly take one step clockwise and one step counterclockwise from q_j in the boundary of the cell. The total number of steps is at most twice the number of edges in the *smaller* of the two new cells that result after the split. If a walk takes more than $O(\log n)$ steps, then we charge each edge of the smaller new cell $O(1)$ time cost for the walk. Since these edges end up in a cell that is at most half the size as before, each edge is charged at most $O(\log n)$ times for walking along it throughout the whole algorithm.

After building the trees inside all Voronoi cells, we combine them and extend them to one tree using the $O(n \log n)$ time algorithm described in [14]. 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

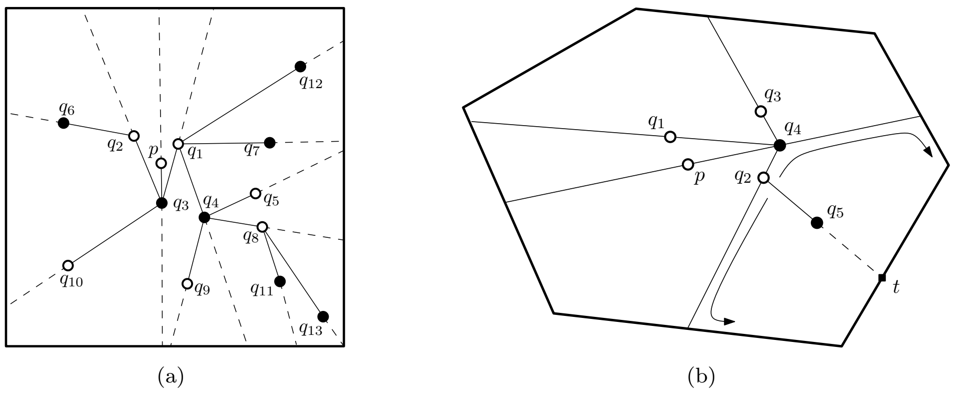

Figure 8: (a) An example of convex cells. (b) After inserting a new point, we subdivide its cell using a tandem walk.

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 2, this is a factor $O(\sqrt{n})$ approximation of the optimum.

Theorem 4 *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 $O(n \log n \log \log n)$ time.*

5 Planar bichromatic minimum spanning trees with k red points

In this section we present an algorithm to compute a planar bichromatic minimum spanning tree of k red points and $n - k$ blue points. The special case when $k = 2$ can be solved by a simple algorithm in $O(n \log n)$ time [7]. For the general case with $k > 2$ red points we present a more involved algorithm that runs in polynomial time when k is constant. Any straightforward algorithm seems to take time exponential in n.

First, we argue that the k red points will be connected to each other in the optimal tree via a number of blue points, at most $k-1$. These are the blue points of degree at least 2 in the optimal tree. We call this subtree of the optimal tree the *skeleton tree.* There are a total of $k(n-k)$ edges between red and blue points, and any skeleton tree is a subset of at most $2k-2$ of them, so the total number of possible skeleton trees is $O((nk)^{2k-2})$. We try all of them (ignoring the non-planar ones).

Given a skeleton tree of the k red points and $O(k)$ blue points, and a set of $O(n)$ unconnected blue points, we want to compute the optimal spanning tree of all points. For all unconnected blue points, we are interested in the set of red points that are still visible from that point, and also in the order of the red points by distance. We compute the set of lines through any pair of points in the given skeleton tree, of which there are $O(k^2)$, and the set of bisectors of any pair of red points, of which there are also $O(k^2)$. These lines together form an arrangement of complexity $O(k^4)$.

Lemma 4 *Within any cell in the arrangement, the ordered list of visible red points is the same for any blue point in that cell.*

Proof: Suppose there are two points p and q in the plane such that some red point r is visible to p, but not to q. This means that there is some fixed edge $\overline{st}$ between r and q that is not between r

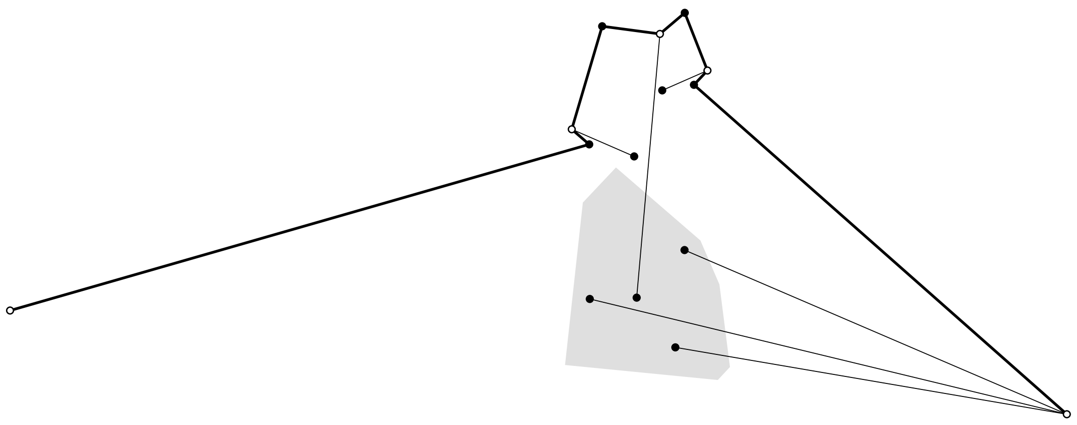

Figure 9: The fat edges show the fixed spanning tree of the five red points and four of the blue points. The thin edges show the optimal minimum planar spanning tree of the given fat tree and the six remaining blue points. The shaded area is a cell of the arrangement.

and p. Therefore either $\overline{st}$ passes between p and q, or the line through r and s or t passes between p and q. In both cases, p and q are not in the same cell.

Suppose two red points r and s are visible to both p and q, but for p r is closer than s, while for q s is closer than r. Then p and q are on different sides of the bisector of r and s, and therefore not in the same cell. ⊠

Now, we would like to connect the blue points within one cell κ directly to the closest red point. However, this may cause intersections with edges for different cells, so some points will need to be connected to other red points. In the optimal solution, the set of points inside each cell will be partitioned into groups that connect to the same red point. These groups are not necessarily convexly separated (at least not given a fixed connection of the red points). In Figure 9 we see a cell with four blue points in it, one of which is connected to the closest red point and three of which are connected to the second closest red point. However, they cannot be separated by a straight line.

Given a cell κ, let B_κ be the blue points inside κ. We will define $\mathbf{r} = \langle r_1, r_2, \ldots, r_{k'} \rangle$ (where $1 \leq k' \leq k$) as the ordered list of visible red points from within κ, where r_1 is the closest red point for all blue points in B_κ. We will show that the subset of B_κ that is connected to r_1 in the optimal solution can be defined by using only three (or fewer) red-blue edges.

Lemma 5 *Assume that no three points are collinear. Consider an optimal spanning tree T, a cell κ in the arrangement induced by the skeleton tree of T, the set B_κ of blue points inside κ, and the ordered list of visible red points $\mathbf{r} = \langle r_1, r_2, \ldots, r_{k'} \rangle$ (where $1 \leq k' \leq k$). Then r_1 is connected to one point of B_κ, or to a subset of B_κ that are exactly those points in the intersection of two or three halfplanes. Two of these halfplanes are bounded by a line through r_1 and a blue point in B_κ, while the third halfplane, if present, is bounded by a line through a different red point and a different blue point not necessarily from B_κ.*

Proof: Assume that r_1 is connected to at least two blue points of B_κ in the optimal spanning tree T. Then, seen from r_1, one blue point b_1 is most counterclockwise of these, and another blue point b_2 is most clockwise of these. Obviously, all other points from B_κ that are connected to r_1 are in the wedge W defined by the intersection of two halfplanes defined by r_1 and b_1, and r_1 and b_2.

Observation 1: No red point lies inside W. The reason is that lines between all pairs of red points

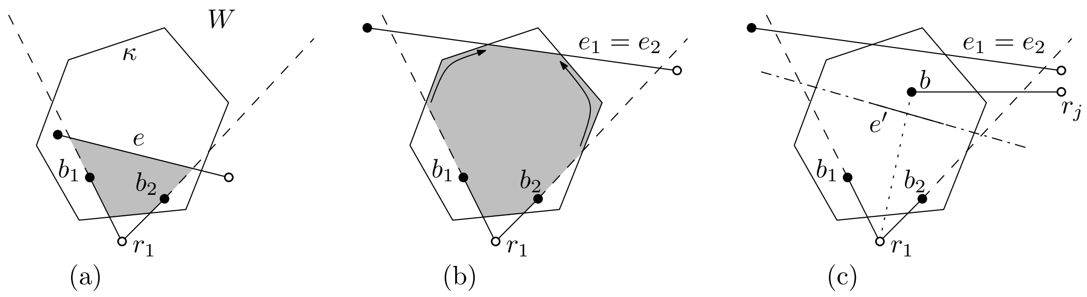

Figure 10: Cases (i) and (ii) of the proof of Lemma 5. The grey regions show the intersection of κ with the three halfplanes.

define the arrangement of which κ is a cell, so no line through r_1 and another red point can intersect κ.

Consider all edges of T that intersect $W \cap \kappa$. We analyze how these can lie.

Case (i): If the extensions $\overrightarrow{r_1b_1}$ beyond b_1 and $\overrightarrow{r_1b_2}$ beyond b_2 hit the same edge e of T first and inside κ (see Figure 10(a)), then the supporting line of e defines a halfplane (to the side of r_1), and all points of B_κ in W and in this halfplane are connected to r_1 in T. If this were not the case, we have a contradiction with the choice of e as the first edge hit, or the planarity of T.

Case (ii): If one or both of the extensions reach the boundary of κ before hitting any edge of T, then we follow the boundary of κ inside W until we find the first edges e_1 and/or e_2 of T that intersect $W \cap \kappa$ (see Figure 10(b)). So from the intersection point of $\overrightarrow{r_1b_1}$ and the boundary of κ we go clockwise and from the intersection point of $\overrightarrow{r_1b_2}$ and the boundary of κ we go counterclockwise. If e_1 and e_2 do not exist, then only two halfplanes define the subset of B_κ of blue points connected to r_1, and we are done. If $e_1 = e_2$ (see Figure 10(b)), then we need the following observation.

Observation 2: For any edge e of T that intersects κ, that has its blue endpoint b_e in W, and its red endpoint is not r_1, there must be another edge e' in T that intersects the line segment $\overline{b_er_1}$. The reason is that $|\overline{b_er_1}| < |e|$, because e intersects κ and for any point inside κ, r_1 is the closest visible red point. Furthermore, the red endpoint of e' must be outside W by Observation 1.

If we apply this observation to any edge $\overline{br_j}$ (with $j > 1$) in T for which $b \in W \cap \kappa$, but $\overline{br_1}$ does not intersect e_1, then one or more edges exist in T that intersect $\overline{br_1}$. Let e' be the one that intersects $\overline{br_1}$ closest to r_1 (see Figure 10(c)). Since e' must cross the boundary of κ (since the red endpoint of e' lies outside $W \cap \kappa$) without intersecting any of $\overline{r_1b_1}$, $\overline{r_1b_2}$, and $e_1 = e_2$, we get a contradiction with the definition of e_1 and e_2. Hence, if b lies as specified, then it is connected to r_1 in T.

It follows that we are in a case similar to case (i). The same two or three halfplanes define the subset of points of B_κ that must be connected to r_1 in T.

Case (iii): $e_1 \neq e_2$, where edge e_1 is the first edge hit in κ by $\overrightarrow{r_1b_1}$, or the first edge intersecting the boundary of κ from the point where $\overrightarrow{r_1b_1}$ hits it in clockwise direction, and edge e_2 is defined similarly with respect to $\overrightarrow{r_1b_2}$, but by going counterclockwise.

By Observation 2, the blue endpoint b_{e_1} of e_1 is not in W, or at least one edge e'_1 in T exists that intersects $\overline{b_{e_1}r_1}$. In the latter case, let e'_1 be that edge of T whose intersection with $\overline{b_{e_1}r_1}$ is closest to r_1. Similarly, we apply the observation to e_2, and if b_{e_2} lies inside W, we define edge e'_2 in T analogously.

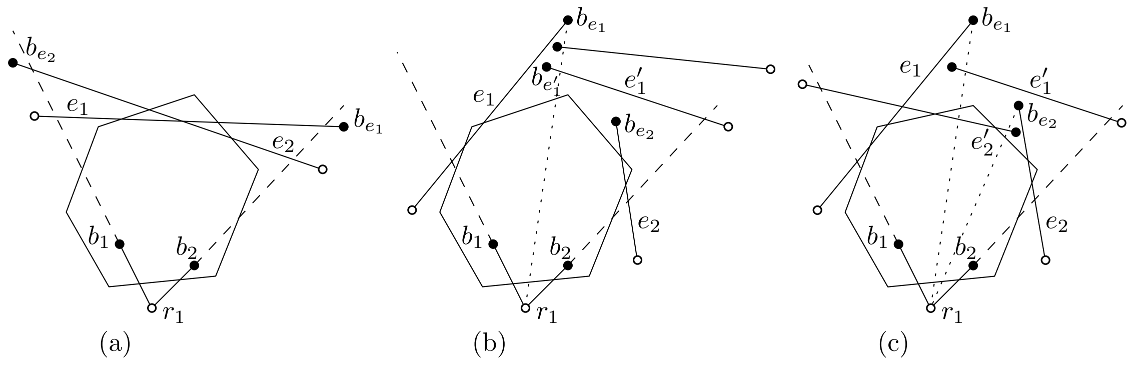

Figure 11: Case (iii) of the proof of Lemma 5.

Assume that b_{e_1} and b_{e_2} are both not in W (see Figure 11(a)). Then we immediately get a contradiction with the definition of e_1 and e_2, or with the planarity of T. So at least one of b_{e_1} and b_{e_2} is in W; assume without loss of generality that it is b_{e_1}. Then e'_1 exists by Observation 2 (see Figure 11(b)).

We claim that the blue endpoint $b_{e'_1}$ of e'_1 lies in W and $\overline{b_{e'_1} r_1}$ does not intersect any edge of T. The red endpoint $r_{e'_1}$ does not lie in W (Observation 1), and if $b_{e'_1}$ would also lie outside W, edge e'_1 contradicts the definition of e_1. So $b_{e'_1}$ lies in W. Now assume that $\overline{b_{e'_1} r_1}$ intersects an edge of T. Then we immediately have a contradiction with the definition of e_1 or e'_1, because the red endpoint of such an edge must again lie outside W (Observation 1). So the claim is true.

Observation 3: The angular interval of W, with r_1 as center, is contained in the union of the angular intervals of e_1 and e'_1. This follows from the fact that $\overline{b_{e_1} r_1}$ intersects e'_1, and $r_{e'_1}$ is outside W so e'_1 intersects $\overrightarrow{r_1 b_2}$. The only possible configuration for e_1 and e'_1 is shown in Figure 11(b).

Since e'_1 intersects $\overrightarrow{r_1 b_2}$, it must do so further from r_1 than e_2 by definition of e_2 (as shown in Figure 11(c)). But then b_{e_2} must also be in W. This implies that e'_2 exists by Observation 2, and hence Observation 3 must hold for e_2 and e'_2 as well. By symmetry e'_2 intersects $\overrightarrow{r_1 b_1}$ further from r_1 than e_1 by definition of e_1, and hence the existence of e'_1 and e'_2 and the way they lie by Observation 3 leads to a contradiction with the planarity of T.

Since all subcases of case (iii) lead to a contradiction, we conclude that $e_1 = e_2$. We already concluded in cases (i) and (ii) that if $e_1 = e_2$, the lemma is true. ⊠

Lemma 5 can be used to generate all possible assignments of subsets of B_κ to the red points. We will guess the at most three edges for T that define the subset of points that are connected to r_1, remove the corresponding blue points from B_κ, then guess the at most three edges for T that define the subset of points that are connected to r_2, and so on. In total, we need to make $O(k)$ guesses of edges in T to connect all points in B_κ to some red point. The collection of edges to choose from are the $O(nk)$ red-blue edges. Hence, we will try $O(nk)^{O(k)} = n^{O(k)}$ partitions of B_κ.

The whole algorithm to compute the planar bichromatic minimum spanning tree is as follows. Choose the skeleton tree with all red points, and all blue points of degree at least 2, by trying all possible subsets of at most $2k-2$ edges from the collection of $O(nk)$ red-blue edges. Choices that do not give a possible skeleton tree are ignored. Every other choice induces a partition of the plane by $O(k^2)$ lines into $O(k^4)$ cells. For each cell, we try all possible assignments of subsets of B_κ to red points, as in Lemma 5. Each resulting spanning tree is tested for planarity and if this is the case, its length is determined. The overall shortest one is the final answer.

We test $O(nk)^{O(k)} = n^{O(k)}$ skeleton trees in $n^{O(k^5)}$ time each, so the total algorithm takes $n^{O(k^5)}$

time.

Theorem 5 *A planar bichromatic minimum spanning tree of k red points and $n-k$ blue points can be computed in $n^{O(k^5)}$ time.*

6 Conclusions and open problems

In this paper we presented several results related to computing a planar bichromatic minimum spanning tree of a set of n red and blue points in the plane. An interesting open problem is whether a constant factor approximation algorithm for the general problem exists that runs in polynomial time. Another open problem is computing the planar bichromatic minimum spanning tree for a constant number of red points and n blue points much more efficiently.

References

[1] M. Abellanas, J. Garcia-Lopez, G. Hernández-Peñalver, M. Noy, and P.A. Ramos. Bipartite embeddings of trees in the plane. *Discr. Appl. Math.*, 93:141–148, 1999.

[2] P. K. Agarwal. Partitioning arrangements of lines, II: Applications. *Discr. Comput. Geom.*, 5:533–573, 1990.

[3] P. K. Agarwal, H. Edelsbrunner, and O. Schwarzkopf. Euclidean minimum spanning trees and bichromatic closest pairs. *Discr. Comput. Geom.*, 6:407–422, 1991.

[4] S. Arora and K. L. Chang. Approximation schemes for degree-restricted MST and red-blue separation problems. *Algorithmica*, 40:189–210, 2004.

[5] M. J. Atallah and D. Z. Chen. On connecting red and blue rectilinear polygonal obstacles with nonintersecting monotone rectilinear paths. *Int. J. Comput. Geom. Appl.*, 11:373–400, 2001.

[6] H. Baumgarten, H. Jung and K. Mehlhorn. Dynamic Point Location in General Subdivisions. *J. Algorithms*, 17(3):342–380,1994.

[7] M. G. Borgelt, M. van Kreveld, M. Löffler, J. Luo, D. Merrick, R. I. Silveira and M. Vahedi. Planar Bichromatic Minimum Spanning Trees. *Technical Report UU-CS-2007-044*, Department of Information and Computing Sciences, Utrecht University, 2007.

[8] J.-D. Boissonnat, J. Czyzowicz, O. Devillers, J. Urrutia and M. Yvinec. Computing Largest Circles Separating Two Sets of Segments. *Int. J. Comput. Geom. Appl.*, 10:41–53, 2000.

[9] P. Bose and G. Toussaint. Growing a tree from its branches. *J. Algorithms*, 19:86–103, 1995.

[10] E. D. Demaine, J. Erickson, F. Hurtado, J. Iacono, S. Langerman, H. Meijer, M. H. Overmars and S. Whitesides. Separating Point Sets in Polygonal Environments. *Int. J. Comput. Geom. Appl.*, 15:403–420, 2005.

[11] H. Everett, J.-M. Robert and M. J. van Kreveld. An optimal algorithm for the ($\leq k$)-levels, with applications to separation and transversal problems. *Int. J. Comput. Geom. Appl.*, 6:247–261, 1996.

[12] M. Grantson, H. Meijer, and D. Rappaport. Bi-chromatic minimum spanning trees. In *Proc. 21st Eur. Workshop on Comput. Geom. (EWCG05)*, pages 199–202, 2005.

[13] M. Hoffmann and C. Tòth. Pointed and colored binary encompassing trees. In *Proc. 21st Annu. ACM Sympos. Comput. Geom.*, pages 81–90, 2005.

[14] F. Hurtado, M. Kano, D. Rappaport and C. D. Tòth. Encompassing colored planar straight line graphs. *Comput. Geom. Theory Appl.*, 39:14–23, 2008.

[15] F. Hurtado, M. Noy, P. A. Ramos and C. Seara. Separating objects in the plane by wedges and strips. *Discr. Appl. Math.*, 109:109–138, 2001.

[16] A. Kaneko and M. Kano. Discrete geometry on red and blue points in the plane – a survey. In *Discrete and Computational Geometry, The Goodman-Pollack Festschrift*, Springer-Verlag, 551–570, 2003.

[17] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. In *Proc. Am. Math. Soc.,* 7:48–50, 1956.

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

[19] H. G. Mairson and J. Stolfi. Reporting and counting intersections between two sets of line segments. In R. A. Earnshaw, editor, *Theoretical Foundations of Computer Graphics and CAD*, volume F40 of NATO ASI, Springer-Verlag, 307–325, 1988.