Geometric Planar Networks on Bichromatic Collinear Points*

Sayan Bandyapadhyay[1], Aritra Banik[2], Sujoy Bhore[3], and Martin Nöllenburg[4]

[1] Department of Informatics, University of Bergen, Norway
sayangcelt@gmail.com
[2] School of Computer Sciences, NISER, Bhubaneswar, India
aritrabanik@gmail.com
[3] Indian Institute of Science Education and Research, Bhopal, India
sujoy.bhore@gmail.com Algorithms and Complexity Group, Technische Universität Wien, Austria
[4] noellenburg@ac.tuwien.ac.at

Abstract. We study three classical graph problems – Hamiltonian path, minimum spanning tree, and minimum perfect matching on geometric graphs induced by bichromatic (red and blue) points. These problems have been widely studied for points in the Euclidean plane, and many of them are NP-hard. In this work, we consider these problems for collinear points. We show that almost all of these problems can be solved in linear time in this setting.

Keywords: Hamiltonian path, minimum spanning tree, minimum prefect matching, red-blue points, collinear points.

1 Introduction

In this article, we study three classical graph problems on geometric graphs induced by bichromatic (red and blue) points. Suppose, we are given a set R of n red points and a set B of m blue points in the Euclidean plane. Consider the complete bipartite graph $G(R, B, E)$ on $R \cup B$, where the set E of edges contains all bichromatic edges between the red points and the blue points. Also, suppose the graph $G(R, B, E)$ is embedded in the plane: the points are the vertices and each edge is represented by the line segment between the two corresponding endpoints. We denote these edges as *bichromatic segments*, where each bichromatic segment connects a red point with a blue point. A subgraph G' of $G(R, B, E)$ (or

* A preliminary version of this paper appeared in the proceedings of 6th Annual International Conference on Algorithms and Discrete Applied Mathematics (CALDAM 2020), and the full version appeared in the Theoretical Computer Science (TCS) journal. The research of Sayan Bandyapadhyay is partly funded by the European Research Council (ERC) via grant LOPPRE, reference 819416. Research of Sujoy Bhore and Martin Nöllenburg is supported by the Austrian Science Fund (FWF) grant P 31119.

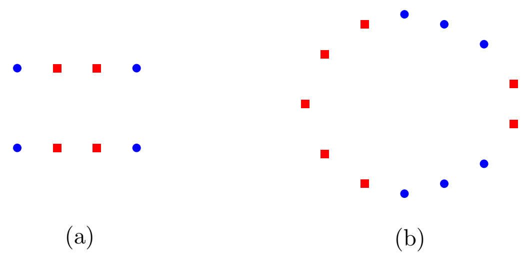


Fig. 1: Two instances without a non-crossing Hamiltonian path. Figure (a) is not in general position, Figure (b) (borrowed from [17]) is in general position.

equivalently a subset of edges of E) is called *non-crossing* (or planar) if no pair of the edges of G' cross each other. Next, we discuss the three graph problems on the bipartite graph $G(R, B, E)$ that we study in this paper.

In the Bichromatic Hamiltonian path problem, the objective is to find a path in $G(R, B, E)$ that spans all the red and blue points. Equivalently, one would like to find a polygonal chain that connects all the red points and the blue points alternately through bichromatic segments. It is not hard to see that a Hamiltonian path exists in $G(R, B, E)$ if and only if $m - 1 \leq n \leq m + 1$, and if there exists one, it can be computed efficiently, as $G(R, B, E)$ is a complete bipartite graph. A more interesting problem is the Non-crossing bichromatic Hamiltonian path problem where the objective is to find a non-crossing Hamiltonian path. Note that one can construct instances with $m - 1 \leq n \leq m + 1$, for which it is not possible to find any non-crossing Hamiltonian path. In Figure 1, we demonstrate two such instances.[5] Figure 1(a) has eight points, where four of them lie on a horizontal line L and the remaining four lie on a line parallel to L. Notice that there must be one red and one blue point with degree 1. One can verify by enumerating all possible paths that there is no non-crossing Hamiltonian path that spans these points. The example in Figure 1(b) has thirteen points in general position, i.e., no three points are collinear, and it also does not admit a non-crossing Hamiltonian path. Indeed, if $2 \leq n \leq 12$, then for any given $\lfloor n/2 \rfloor$ red (resp. blue) points and $\lceil n/2 \rceil$ blue (resp. red) points in general position, there exists a non-crossing Hamiltonian path [18]. Due to the uncertainty of the existence of non-crossing Hamiltonian paths in the general case, researchers have also considered the problem of finding a non-crossing alternating path of length as large as possible [16,20]. Recently, Mulzer and Valtr [21] showed that for a set of points in convex position, there is an absolute constant $\varepsilon > 0$, independent of n and of the bicoloring of the points, such that P always admits a non-crossing alternating path of length at least $(1 + \varepsilon)n$. Garcia and Tejel [15] designed polynomial time algorithms for special geometric instances.

[5] In all the figures throughout the paper, we show red (resp. blue) points by squares (resp. disks).

Next, we consider the Bichromatic spanning tree problem where the objective is to compute a minimum weight spanning tree of $G(R, B, E)$, where the weight of the edge between each pair of points is the Euclidean distance between those two points. Note that this problem can be solved efficiently by any standard minimum spanning tree algorithm. Biniaz *et al.* [7] showed that minimum bichromatic spanning tree can be computed in $O(n \log n)$ time. Moreover, their algorithm extends to the multicolored version with the time complexity $O(n \log n \log k)$ where k is the number of different colors (or the size of the multipartition in a complete multipartite geometric graph). A more interesting problem is the Non-crossing bichromatic spanning tree problem where additionally the computed tree must be non-crossing. Borgelt *et al.* [10] showed that this problem is NP-hard. For points in general position, they gave a near-linear time $O(\sqrt{n})$-approximation. On the other hand, for points in convex position, they gave an exact cubic-time algorithm. Another line of work that received much attention is where the task is to find a degree-bounded non-crossing spanning tree [9].

Finally, we consider the Bichromatic matching problem. Again assume that $n = m$ for simplicity. We would like to find a minimum weight perfect matching in $G(R, B, E)$. The weight of an edge is the Euclidean distance between its corresponding points. It is a well-known fact that a minimum weight bichromatic matching (for points in general position) in the plane is always non-crossing, which follows from the observation that the sum of the diagonals of a convex quadrilateral is strictly larger than the sum of any pair of opposite sides. This implies that using any standard bipartite matching algorithm one can solve a Non-crossing Bichromatic matching exactly. But, algorithms with better running time have been designed by exploiting the underlying geometry of the plane. Recently, Kaplan *et al.* [19] designed an $O(n^2 \operatorname{poly}(\log n))$ algorithm for the problem improving the $O(n^{2+\epsilon})$ algorithm due to Agarwal *et al.* [2], where poly(.) is a polynomial function. In [21], Mulzer and Valtr showed that for a set of in convex position, there always exists a non-crossing bichromatic separated matching[6] on at least $(1+\varepsilon)n$ points of P, where $\varepsilon > 0$ that is independent of n and of the bicoloring of the points. Abu-Affash *et al.* [1] studied the bottleneck variant of the non-crossing matching problem. This problem is known to be NP-Hard. They gave an $O(n \log n)$-time approximation algorithm which computes a non-crossing matching of size at least $\frac{2n}{5}$ edges, whose edges have length at most $\sqrt{2} + \sqrt{3}$ times the bottleneck. Biniaz *et al.* [8] studied the non-crossing geodesic spanning trees, Hamiltonian cycle, and perfect matching in a simple polygon.

In this article, we consider the above mentioned problems for collinear points on a real line[7]. We assume that the points are given in their sorted order. We note that the case of non-crossing graphs on collinear points is closely related to 1-page or 2-page book embeddings [6], which have all vertices placed on a line (called the spine) and the edges drawn without crossings in one or two of

[6] This is a properly colored matching whose segments are pairwise disjoint and intersected by common line.

[7] Throughout the paper we assume that all points are distinct.

the halfplanes (but not in both) defined by the spine (called the pages). In our case we assume that the edges are drawn as (circular) arcs or 1-bend polylines either above or below the spine.[8] We assume that their weight is given by the Euclidean distance of their endpoints. If the arcs are drawn infinitesimally close to the spine, these weights correspond to the lengths of the arcs.

1.1 Our Results

The main results obtained in this work are the following.

- Non-crossing Hamiltonian path for collinear points – We prove that for any collinear configuration of the points with $|R| = |B|$, there always exists a non-crossing Hamiltonian path. Additionally, we give a linear-time algorithm for computing such a path (Section 2). As noted before, if the points are not collinear and lie in the plane, the existential claim is not necessarily true.
- Minimum spanning tree for collinear points – We give a linear-time algorithm for computing a minimum-weight spanning tree, and a quadratic-time algorithm for computing a minimum-weight non-crossing spanning tree for collinear points and edges on a single page (Section 3). In contrast, the problem in the non-crossing case is NP-Hard in the plane.
- Minimum non-crossing matching for collinear points – We give a linear-time algorithm that computes a minimum-weight non-crossing perfect matching for collinear points and edges on a single page (Section 4). We note that the most efficient algorithm for this problem in the plane runs in $O(n^2 \operatorname{poly}(\log n))$ time.

We note that even in this simple one-dimensional case these problems become sufficiently challenging if one is constrained to use only linear (or near-linear) time. Our study of one-dimensional case is partly motivated due to the intractability of the problems or lack of efficient algorithms in the plane. Note that for all the problems, we have obtained improved or more interesting results compared to the planar case. Our findings give a better understanding of these problems and our work can be considered as a stepping stone towards achieving improved results in the plane.

Remark. In a parallel work, Aichholzer et al. [3] obtained a different linear time algorithm for Non-crossing Hamiltonian path for collinear points.

1.2 Related Work.

A closely related problem to Bichromatic Hamiltonian path is the Bichromatic traveling salesman problem (Bichromatic TSP) problem, where one would like to find a minimum-weight Hamiltonian cycle in $G(R, B, E)$. The

[8] For the sake of convenience, we draw edges as simple curves in all the figures.

weight of each edge is the (Euclidean) length of the corresponding segment. The weight of a path is the sum of the weights of the edges along the path. We assume $n = m$, otherwise, there is no Bichromatic Hamiltonian cycle. A straightforward reduction from the (monochromatic) EUCLIDEAN TSP [5] (replace each point by a bichromatic pair that is small distance apart) shows that BICHROMATIC TSP is also NP-hard. One simple, but powerful fact is that an optimum Euclidean TSP is always non-crossing. This helps to obtain a PTAS [5] for the problem. However, an optimum Bichromatic TSP is not necessarily non-crossing which makes its computation much harder compared to Euclidean TSP. The best known approximation factor for the Bichromatic TSP problem is 2 due to Frank *et al.* [14] who improved the 2.5-approximation of Anily *et al.* [4]. For a set of collinear points, Evans *et al.* [13] gave a quadratic time algorithm for computing an optimum non-crossing TSP, every edge of which is a poly-line with at most two bends. We note that, in the existing research literature, the framework used by Evans *et al.* [13] is the closest to our work.

Colannino et al. [11] gave a near-linear time algorithm for the many-to-many matching problem for red-blue points on a line, which is similar to bichromatic matching. In many-to-many matching, we are given a bipartite graph, and we want to find a minimum-weight subset of edges that span all the vertices. Note that in contrast to our problem, here two edges can share an endpoint.

2 Non-crossing Hamiltonian Path for Collinear Points

If we would require for the collinear point set that each edge of a Hamiltonian path is a straight-line segment, the problem becomes trivial: an input instance can have a non-crossing Hamiltonian path if and only if the colors of the points alternate. Therefore, we consider the case where edges are represented by circular arcs drawn in the halfplane either above or below the line.

Definition 1. ***Non-crossing Hamiltonian path for collinear points****. Given a set R of n red points and a set B of n blue points on a line, find a non-crossing geometric path π in the plane such that π consists of a sequence of circular arcs above or below H, each of which connects a red and a blue point and π spans all the input points.*

Note that in the above definition if the path is allowed to use arcs only from above (resp. below) H, then there might not exist such a Hamiltonian path. For example, consider the configuration in Figure 2, which contains 16 points. Also, consider the continuous monochromatic chunks of points. Thus, the first chunk contains four blue points, the second chunk contains two red points, and so on. Note that the first blue point must be connected to the last red point in any non-crossing Hamiltonian path. Otherwise, there are two options: it gets connected to (i) a different point of the fourth chunk and (ii) a point of the second chunk. In the first case, the last red point cannot get connected to a blue point. In the second case, at least one point of the first chunk cannot get connected to a red point. Thus, the first blue point must be connected to the last red point.

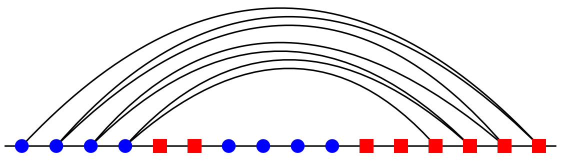


Fig. 2: Example of a set of collinear points for which a non-crossing Hamiltonian path does not exist if the arcs can be drawn only above the spine.

Moreover, for similar reasons, the second blue point must be connected to the second to last red point, the third blue point must be connected to the third to last red point and the fourth blue point must be connected to the fourth to last red point. Figure 2 shows such an example Hamiltonian path. It is easy to verify that this path cannot be completed by connecting all the points.

In the light of the above discussion, we would like to find a non-crossing Hamiltonian path each of whose edges is a circular arc that lies either above or below H. First, we give a simple and intuitive construction of such a path for any configuration of points. The construction itself takes polynomial time, hence giving a polynomial-time algorithm for computation of such a path. Later, we give a more involved algorithm that runs in linear time.

2.1 The Construction

To construct the path, we start with any bichromatic matching (not necessarily non-crossing) of the points. Note that each matching edge is a segment on H. We will connect these edges to obtain a Hamiltonian path. First, we form a hierarchical (or laminar) structure of these matching edges. Informally, the matching edges are hierarchical if any two edges are either separated or one is nested in the other.

Definition 2. *A set of matching edges M are hierarchical (or laminar) if for any two edges $(u, v), (w, x) \in M$ with $u < v$, $w < x$ and $u < w$, either $u < v < w < x$ ($(u, v), (w, x)$ are separated) or $u < w < x < v$ ((w, x) is nested in (u, v)).*

Given any matching M for $R \cup B$, we can change it to a hierarchical matching in the following way. If there are two edges $(u, v), (w, x) \in M$ with $u < v$, $w < x$, $u < w$ that are not separated and none of them is nested in the other, then it must be the case that $u < w < v < x$. Now, there are two subcases, depending on the colors of u and w. If u, w are red or u, w are blue, we replace the edges $(u, v), (w, x)$ by the two bichromatic edges $(u, x), (w, v)$. Otherwise, either u is red, w is blue or u is blue, w is red. In that case, we replace the edges $(u, v), (w, x)$ by the two bichromatic edges $(u, w), (v, x)$. See Figure 3 for a demonstration. Note that in all the cases, the new pair of edges does not violate the hierarchical structure. We repeat the process for each pair of edges that violates the condition. Newly formed edges might violate the condition with respect to other edges. However, if an edge is removed, it is never added back,

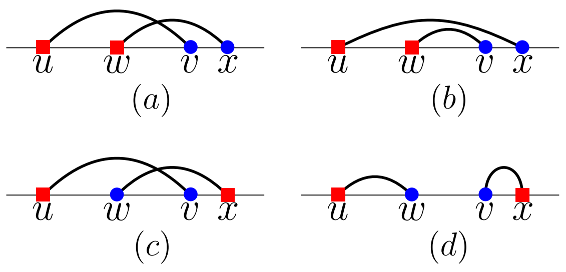


Fig. 3: Illustration showing uncrossing of edges in the first ((a),(b)) and second subcases ((c),(d)). (a) and (c) showing edges before uncrossing. (b) and (d) showing edges after uncrossing.

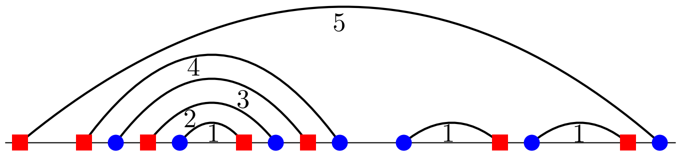


Fig. 4: Illustration showing the levels of the edges.

and thus the process will eventually stop at some point when no pair of edges violates the condition any more.

Next, we associate levels with each matching edge of M in a recursive way. In the base case, for each edge that does not nest any other edge, set its level to 1. Now, suppose we have defined edges of level j for each $j \leq i-1$ for $i \geq 2$. An edge (u, v) has level i, if it nests a level $i-1$ edge, and for any level $i-1$ edge (w, x) that it nests, there is no other edge that is nested in (u, v) that also nests (w, x) (see Figure 4). Note that the level of each edge is unique. Let L be the maximum level.

For any edge (u, v) of M with level j, call the points that lie between u and v including u and v as a level j block. Thus, a level l block is a union of disjoint blocks of levels at most $l-1$ and two special points which are the first and last point of the block.

Observation 1 *Each block as defined above contains the same number of red and blue points.*

Proof. We prove this by induction on the level l of the blocks. In the base case, $l = 1$ and each level 1 block consists of the two endpoints of the corresponding edge. So, the statement holds in this case. Now, suppose the statement is true for all blocks with level $l \leq j$. Consider any block β of level $j+1$ corresponding to the edge (u, v). Now, let β be the union of the disjoint blocks $\beta_1, \ldots, \beta_t$ along with the points u, v. As each β_i has level at most j, by induction, it has the same number of red and blue points. As (u, v) is bichromatic, it follows that β as well has the same number of red and blue points. □

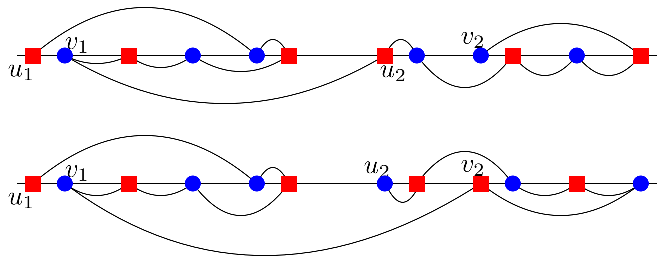


Fig. 5: Illustration showing Case 1 (upper) and Case 2 (lower) of Lemma 2.

We compute the Hamiltonian path for all the blocks in a bottom up manner. The path of a level 1 block is the matching edge itself which defines the block. Additionally, for each block, we compute a path for the block that satisfies the following two invariants.

- The first point of the block is an endpoint of the path.
- If an endpoint p of the path is not an endpoint of the block, then the path cannot contain two edges $(u, v), (w, x)$ with $u < v$ and $w < x$, such that (u, v) lies above H, (w, x) lies below H, $u < p < v$, and $w < p < x$.

Informally, the second condition states that the endpoint of the path that is not an endpoint of the block should be available for connecting with an edge at least from one side. Note that the paths for level 1 blocks trivially satisfy the invariants. Now, assume that we have computed the paths for all the level j blocks for $j \leq l - 1$ and $l \geq 2$ that satisfy the invariants. We show how to compute the path for a level l block S that also satisfies the invariants. Let u, v be the endpoints of the block. Also let $S_1, \ldots, S_t$ be the blocks, sorted w.r.t the index of the first point in increasing order, whose union with the set $\{u, v\}$ forms the block S. As S_i has level at most $l - 1$, we have already computed the path of S_i for all i. We show, by induction, how to construct the path T' for the points in $\cup_{j=1}^{i} S_j$ for all $2 \leq i \leq t$. Then, we show how to join the edge (u, v) with T' to obtain the path for the block S. For simplicity, we also refer to the set of points $\cup_{j=1}^{i} S_j$ as a block. Now, we prove the following lemma.

Lemma 2. *A non-crossing Hamiltonian path of $\cup_{j=1}^{i} S_j$ can be computed for all $1 \leq i \leq t$ that satisfies the two invariants.*

Proof. We prove this using induction on the values of i. In the base case, for $i = 1$, we know how to compute the path of $\cup_{j=1}^{i} S_j = S_1$ that satisfies the two invariants. Now, consider any $i \geq 2$. Suppose we have already computed the path T_{i-1} of $\cup_{j=1}^{i-1} S_j$ that satisfies the two invariants. Let Π_i be the path of S_i that also satisfies the two invariants. Let u_1, v_1 (resp. u_2, v_2) be the endpoints of the path T_{i-1} (resp. Π_i) with $u_1 < v_1$ (resp. $u_2 < v_2$). As $\cup_{j=1}^{i-1} S_j$ (resp. S_i) contains the same number of red and blue points, the color of u_1 (resp. u_2) will be different from the color of v_1 (resp. v_2). Now, there are two cases.

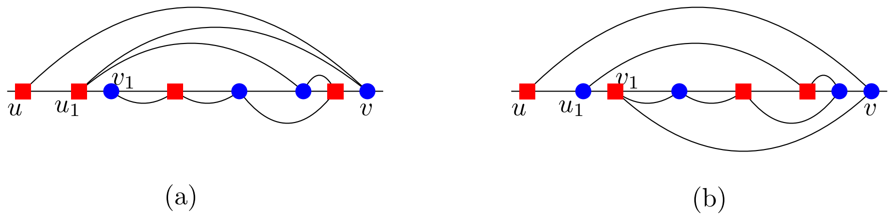


Fig. 6: (a) Illustration showing Case 1 of Lemma 3. (b) Illustration showing Case 2 of Lemma 3.

1. *u_1 and u_2 have the same color.* We add the edge (v_1, u_2) with $T_{i-1} \cup \Pi_i$ to get the path T_i for $\cup_{j=1}^{i} S_j$ (see Figure 5). To make sure (v_1, u_2) does not cross the other edges, we can make use of the second invariant. From the invariant it follows that, for v_1, all the edges of T_{i-1} whose endpoints are on both sides of v_1 must lie on the same side of H. Thus, if those edges lie above, we draw the edge (v_1, u_2) below H. Otherwise, we draw (v_1, u_2) above H. Note that u_1 is an endpoint of T_i which is the first point of $\cup_{j=1}^{i} S_j$. Also $v_1 < u_2 < v_2$. Thus, the other endpoint v_2, still satisfies the second invariant as before by induction.

2. *u_1 and u_2 have different colors.* We add the edge (v_1, v_2) with $T_{i-1} \cup \Pi_i$ to get the path T_i for $\cup_{j=1}^{i} S_j$ (see Figure 5). We need to ensure that the edges of T_{i-1} and Π_i whose endpoints are on both sides of v_1 and v_2, respectively, lie on the same side of H. If this is not true, the edges of Π_i that lie below H can be redrawn above H, and the edges of Π_i that lie above H can be redrawn below H. This does not violate any invariant. Hence, (v_1, v_2) can be drawn without crossing any edge of $T_{i-1} \cup \Pi_i$. Note that u_1 is an endpoint of T_i which is the first point of $\cup_{j=1}^{i} S_j$. The other endpoint u_2 was an endpoint of the block S_i. Thus, even after the drawing of (v_1, v_2) one side of u_2 still remains available. Hence, the second invariant is also satisfied. □

The next lemma completes the induction step for showing the construction of the path for the level l block.

Lemma 3. *A non-crossing Hamiltonian path for the level l block S can be computed that satisfies the two invariants.*

Proof. First, we compute the path T_t for the points in $\cup_{j=1}^{t} S_j$ using the construction in Lemma 2. Let u_1, v_1 be the endpoints of T_t such that $u_1 < v_1$. Note that as mentioned before $S = (\cup_{j=1}^{t} S_j) \cup \{u, v\}$. Without loss of generality, assume the color of u and v is red and blue, respectively. The other case can be handled similarly. Now, there can be two cases.

1. *u_1 is red and v_1 is blue.* We add the edge (u_1, v) with $T_t \cup \{(u, v)\}$ to get the path for S (see Figure 6(a)). From the second invariant for T_t it follows that, for v_1, all the edges of T_t whose endpoints are on both sides of v_1 must lie on the

same side of H. Thus, if those edges lie above, we draw the edges $(u_1, v), (u, v)$ above H. Otherwise, we draw $(u_1, v), (u, v)$ below H. Thus, the second invariant is satisfied. Also, note that u is an endpoint of the new path which is the first point of S. Hence, both the invariants are satisfied.

2. *u_1 is blue and v_1 is red.* We add the edge (v_1, v) with $T_t \cup \{(u, v)\}$ to get the path for S (see Figure 6(b)). From the second invariant for T_t it follows that, for v_1, all the edges of T_t whose endpoints are on both sides of v_1 must lie on the same side of H. Thus, if those edges lie above, we draw the edge (v_1, v) below H. Otherwise, we draw (v_1, v) above H. As u_1, an endpoint of the new path, is the second point of S, irrespective of how we draw (u, v), the second invariant is satisfied. Also u is an endpoint of the new path which is the first point of S. Hence, both the invariants are satisfied in this case as well. □

To compute the path of all the points in $R \cup B$ one can note that $R \cup B$ is the union of a set of blocks having levels at most the maximum level L. By Lemma 3, we can compute the paths for all such blocks that satisfy the invariants. Then we can merge those paths using the construction in Lemma 2 to get the path for the points in $R \cup B$. It is easy to verify that the overall construction can be done in polynomial time. Thus, we get the following theorem.

Theorem 1. *For any set R of red points and B of blue points on a line with $|R| = |B|$, there always exists a non-crossing Hamiltonian path whose edges are circular arcs that lie above or below H. Moreover, such a path can be computed in polynomial time.*

2.2 A Linear Time Algorithm for Non-Crossing Hamiltonian Path

In this subsection, we give another algorithm for computing a non-crossing Hamiltonian path. This algorithm uses very different set of ideas than the previous algorithm. Recall that all the input points lie on a line. We assume that the points are given in sorted order with respect to their x coordinates. For a point p (except the last one), let $S(p)$ be the point which is the successor of p in this order. We use the following algorithm to compute a non-crossing Hamiltonian path. In contrast to the previous algorithm, this algorithm processes the points from left to right and extends the Hamiltonian path constructed so far by connecting the current point with an appropriately chosen point. In particular, in every iteration, we consider a point p and connect it by adding one or more edges. Initially, p is the leftmost point. We also maintain a set of active points which is initialized to the set of all points. We store the constructed path in a set of edges Π, which is initially empty.

- Let $\mathsf{Right}(r)$ and $\mathsf{Right}(b)$ be the rightmost (or last in the order) red and blue points, respectively, which are active.
- If the color of p is different from the color of $S(p)$, we simply add an arc $(p, S(p))$ to Π that lies above H. Make p inactive.

- Otherwise, there are two cases.
  (i). If p is red, add two edges $(p, \mathsf{Right}(b))$ and $(\mathsf{Right}(b), S(p))$ to Π. These two edges are drawn above H as circular arcs. Make p and $\mathsf{Right}(b)$ inactive.
  (ii). If p is blue, add two edges $(p, \mathsf{Right}(r))$ and $(\mathsf{Right}(r), S(p))$ to Π. These two edges are drawn below H as circular arcs. Make p and $\mathsf{Right}(r)$ inactive.
- If $S(p)$ is active, assign $S(p)$ to p (i.e., $p \leftarrow S(p)$) and repeat all the steps. Otherwise, terminate the algorithm.

The different iterations of the above algorithm are shown in an example in Figure 7. Now we discuss the correctness of the algorithm. First, we have the following observation.

Observation 4 *Consider any iteration of the algorithm. Then, any red point on the right of $\mathsf{Right}(r)$ (if any) is inactive and has degree 2. Similarly, any blue point on the right of $\mathsf{Right}(b)$ (if any) is inactive and has degree 2. Moreover, any point on the left of p (if any) is inactive and except the first point all of them have degree 2.*

Lemma 5. *The algorithm correctly computes a bichromatic Hamiltonian path.*

Proof. Note that when the algorithm terminates, $S(p)$ is inactive. Thus, its degree must be 2. If $S(p)$ is red (resp. blue), then it had become $\mathsf{Right}(r)$ (resp. $\mathsf{Right}(b)$) at some point and its degree is 2. By Observation 4, all the points whose colors are same as the color of $S(p)$ and lie on the right of $S(p)$ have degree 2. Also, the degree of all the points on the left of p except the first point is 2. The degree of p and the first point is 1. As the number of red and blue points are same, all the points that lie on the right of $S(p)$ must have degree 2. Thus, Π is a subgraph where each vertex has degree 2 except two special vertices whose degrees are 1. It follows that such a subgraph is a path that spans all the vertices. Also, all the edges on this path Π are bichromatic, and hence Π is a valid bichromatic Hamiltonian path. □

Next, we argue that the computed Hamiltonian path is non-crossing. The arcs that are added between points p and $S(p)$ in the second step do not cross any other drawn edges, as p and $S(p)$ are consecutive points. Also, the edges drawn above H do not cross any edges drawn below H. Moreover, the edges $(p, \mathsf{Right}(r))$ and $(\mathsf{Right}(r), S(p))$ (or $(p, \mathsf{Right}(b))$ and $(\mathsf{Right}(b), S(p))$) drawn in the same iteration do not cross each other. The following observation completes the claim.

Observation 6 *Consider two edges (u, v) and (u', v') which are drawn as circular arcs above (resp. below) H and added to Π in different iterations. Then, either (u, v) is nested in (u', v'), or (u', v') is nested in (u, v).*

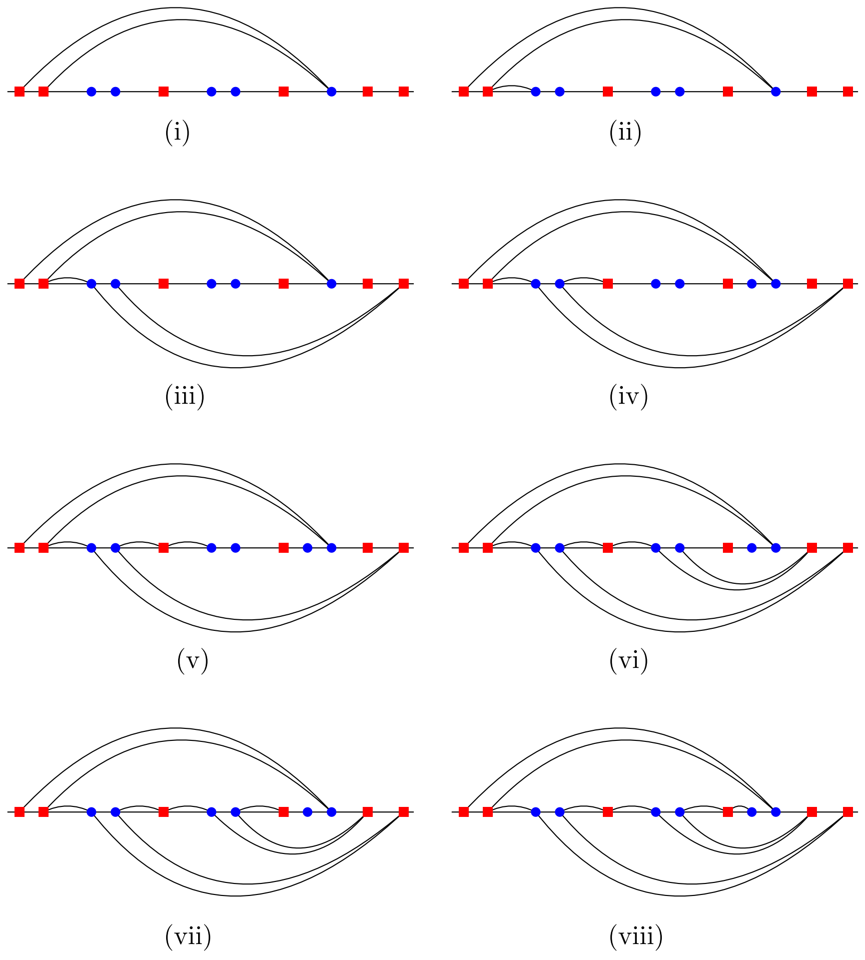


Fig. 7: Figure showing the execution of the Hamiltonian path computation algorithm on an example point set.

The algorithm can be implemented to run in linear time. Note that given the values of p, $\mathsf{Right}(r)$ and $\mathsf{Right}(b)$, each iteration of the algorithm can be performed in $O(1)$ time. Thus, it is sufficient to show that the number of iterations is at most the number of points. We use three pointers to keep track of p, $\mathsf{Right}(r)$ and $\mathsf{Right}(b)$ in each iteration. In every iteration, we set $S(p)$ to be the new p. Thus, the pointer to p always moves from left to right, i.e, it tracks each point at most once. Also, in an iteration, when we change $\mathsf{Right}(r)$ (resp. $\mathsf{Right}(b)$) the red (resp. blue) point on its left becomes the new $\mathsf{Right}(r)$ (resp. $\mathsf{Right}(b)$). Thus, the two pointers to $\mathsf{Right}(r)$ and $\mathsf{Right}(b)$ always move from right to left. Hence, the linear running time of the algorithm follows.

Theorem 2. *For any set R of red points and B of blue points on a line with $|R| = |B|$, a non-crossing Hamiltonian path can be computed in linear time whose edges are circular arcs that lie above or below H.*

3 Minimum Spanning Tree for Collinear Points

In this section, we study the BICHROMATIC SPANNING TREE problem for collinear points. We proceed with the following definition.

Definition 3. ***Spanning tree for collinear points.*** *Given a set R of n red points and a set B of m blue points all of which lie on a line, find a minimum weight geometric tree T in the plane such that each edge of T is represented by a circular arc that lies above H, each arc connects a red and a blue point, and T spans all the input points. The weight of an arc is given by the Euclidean distance of its endpoints. In the non-crossing version of the problem, one would like to compute such a tree so that the corresponding circular arcs are non-crossing.*

First, we discuss a greedy linear time algorithm for computing an optimum, i.e., minimum-weight spanning tree, which potentially has crossings.

3.1 Spanning Tree with Crossings

Let $p_1, \ldots, p_{n+m}$ be the input points sorted in increasing order of their x coordinates. For each point p_i, let $\mathrm{col}(p_i)$ be the color of p_i. We assume that the points are given in this order. Our algorithm has two steps. In the first step, we traverse the points in the sorted order and connect each point with its nearest opposite color point using an arc if it is not already connected. This gives us a set of components $\{C_1, \ldots, C_k\}$ for $k \leq n$, where each component contains at least one edge. For any component $C_i, 1 \leq i \leq k$, let $l(C_i)$ and $r(C_i)$ be the leftmost and the rightmost point, respectively. In the second step, we traverse the components from left to right. Consider the first two components C_1 and C_2. If $\mathrm{col}(r(C_1)) \neq \mathrm{col}(l(C_1))$, join C_1 and C_2 by an arc $(r(C_1), l(C_2))$. Otherwise, check the distance between $r(C_1)$ and its nearest opposite color point in C_2, and the same for $l(C_2)$ and its nearest opposite color point in C_1. Choose the shorter

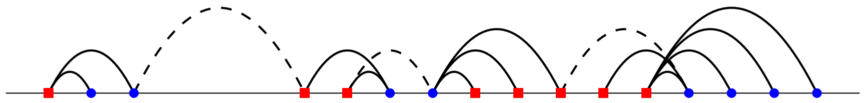


Fig. 8: Figure demonstrating the execution of the algorithm on an example. The dashed arcs are added in the second step.

one to join C_1 and C_2. We repeat the same process for each consecutive pair of the remaining components. See Figure 8 for an example.

Note that after the first step each component C_i is a tree. In the second step, we add exactly one arc between a consecutive pair of components. Hence, the selected arcs form a valid spanning tree. Next, we move on towards proving the correctness of the algorithm. We need a few definitions for that. Consider any graph $G = (V, E)$ and a subset of edges X in this graph. Also consider a subset S of vertices. An edge e is said to cross the cut $(S, V \setminus S)$ or be across the cut $(S, V \setminus S)$ if one of its endpoint is in S and another endpoint is in $V \setminus S$. X does not cross the cut $(S, V \setminus S)$ if none of its edges cross the cut.

To prove the correctness of our algorithm, we need the following standard lemma [12] from the literature of minimum spanning tree.

Lemma 7 (Cut property). *[12] Suppose the set of edges X are part of a minimum spanning tree of $G = (V, E)$. Pick any subset of vertices S for which X does not cross the cut $(S, V \setminus S)$, and let e be a minimum weight edge across this cut. Then $X \cup \{e\}$ is part of some minimum spanning tree.*

Next, we prove the optimality of our algorithm.

Lemma 8. *At any moment, let X be the set of edges added by the algorithm so far. Then there is an optimum bichromatic spanning tree that contains X.*

Proof. We prove this lemma by induction. In the base case, $X = \emptyset$ and the lemma is vacuously true. Now, consider any moment when X is the set of edges added so far by the algorithm such that there is an optimum bichromatic spanning tree that contains X. Let $e = (u, v)$ be the next bichromatic edge added by the algorithm. Now, there can be two cases. e is added in the first or second step.

Suppose e is added in the first step. We show that e is a minimum weight edge across a cut such that X does not cross the edges of this cut. Then the statement of the lemma holds by the Cut property of Lemma 7. Note that in the first step, we connect each point to its nearest opposite color point if it is not already connected. WLOG, suppose (u, v) was added corresponding to u. Then X does not cross the cut $(V \setminus \{u\}, \{u\})$, as u is not yet connected. Also, the way we connect u to its nearest opposite color point, (u, v) must be a minimum weight edge across $(V \setminus \{u\}, \{u\})$ in the (implicit) bichromatic input graph. Thus, the lemma holds in this case.

Next, suppose $e = (u, v)$ is added in the second step. WLOG, let u be in the component C_i and v be in C_{i+1}. Consider the cut $(P_1, V \setminus P_1)$ such that

$P_1 = \cup_{j=1}^{i} C_j$. Then, by the way we select the edge across C_i and C_{i+1}, it follows that (u, v) must be a minimum weight (bichromatic) edge across the cut $(P_1, V \setminus P_1)$. Now, consider any edge in X. If it is added in the first step, it cannot cross $(P_1, V \setminus P_1)$, as it must lie within a component. Otherwise, the edge is added in the second step. But, in this case its endpoints should lie in P_1, as we connect the consecutive components from left to right. By the Cut property, the lemma holds in this case as well. □

Now, we prove the key theorem of this subsection.

Theorem 3. *For any set R of red points and B of blue points on a line, an optimum spanning tree can be computed in linear time.*

Proof. To prove this theorem, we run the above algorithm. The correctness follows by Lemma 8 when the algorithm terminates. As the nearest neighbors (of opposite color) of all the points can be computed and stored in linear time given the sorted points, the algorithm can be executed in linear time. This completes the proof of the theorem. □

Next, we discuss the algorithm for the non-crossing case.

3.2 Non-crossing Spanning Tree

Let $P_1, P_2, \ldots, P_k$ be the alternating monochromatic chunks of points ordered from left to right for $k \in \{1, \ldots, n\}$. Thus, the color of the points in P_i is different from the color of the points in P_{i+1} for all $1 \leq i \leq k-1$. An arc (p, q) is called a consecutive (resp. non-consecutive) arc if it connects points from two consecutive (resp. non-consecutive) chunks. We start with the following observation.

Observation 9 *Consider any point $p \in P_i$. If an arc (p, q) is contained in a minimum spanning tree, then either $q \in P_{i-1}$ or $q \in P_{i+1}$, i.e., (p, q) must be a consecutive arc.*

Proof. Consider any minimum spanning tree T. Suppose there is a non-consecut ive arc in T. Consider an arc (p, q) in T such that (p, q) is a minimal non-consecutive arc, i.e., all arcs nested in (p, q) are consecutive arcs. We claim that it is always possible to replace (p, q) in T by a consecutive arc (p_1, q_1) such that $T_1 = (T \setminus \{(p, q)\}) \cup \{(p_1, q_1)\}$ is a non-crossing spanning tree and the cost of T_1 is strictly less than the cost of T. But, this leads to a contradiction, and hence there cannot be any non-consecutive arc in T. Next, we prove the claim.

Let p' be the rightmost point between p and q (including p) that is connected to p in $T \setminus \{(p, q)\}$ (see Figure 9(a)). Let q' be the point on the immediate right of p'. Note that q' is connected to q in $T \setminus \{(p, q)\}$. If p' and q' are of different colors, then we replace (p, q) in T by the consecutive arc (p', q') (see Figure 9(b)). There is no crossings in $T_1 = (T \setminus \{(p, q)\}) \cup \{(p', q')\}$, as p' and q' are consecutive points. T_1 is a spanning tree, as all the points are contained in it and the two connected components in $T \setminus \{(p, q)\}$ are joined in T_1 by the arc

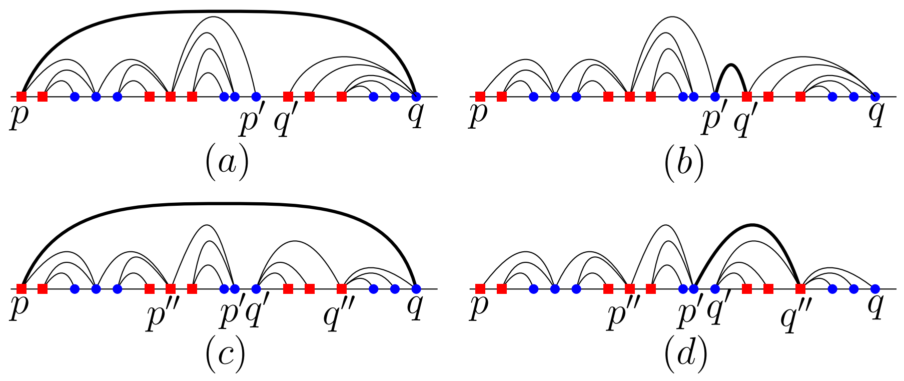


Fig. 9: (a),(b) show the case when p' and q' are of different colors. (c),(d) show the case when p' and q' have same color. (b) shows the tree obtained after replacing (p, q) in (a) by (p', q'). (d) shows the tree obtained after replacing (p, q) in (c) by (p', q'').

(p', q'). As (p', q') is a consecutive arc nested in the non-consecutive arc (p, q), the cost of T_1 is strictly less than the cost of T.

Now, suppose p' and q' have the same color. If $q \neq q'$, consider any arc e_1 in $T \setminus \{(p, q)\}$ which contains q' as an endpoint and there is no other arc in $T \setminus \{(p, q)\}$ containing q' which nests e. Let $e_1 = (q', q'')$ (see Figure 9(c)). The color of p' and q'' are different, as e_1 is bichromatic and p' and q' have the same color. Note that if $q = q'$, there might not be any arc in $T \setminus \{(p, q)\}$ containing q', and thus e_1 cannot be defined. Similarly, if $p \neq p'$, consider any arc e_2 in $T \setminus \{(p, q)\}$ which contains p' as an endpoint and there is no other arc in $T \setminus \{(p, q)\}$ containing p' which nests e_2. Let $e_2 = (p', p'')$ (see Figure 9(c)). The color of q' and p'' are different, as e_2 is bichromatic and p' and q' have the same color. Note that if $p = p'$, there might not be any arc in $T \setminus \{(p, q)\}$ containing p', and thus e_2 cannot be defined. But, at least one of e_1 or e_2 must be defined, as $p = p'$ and $q = q'$ cannot happen at the same time. Otherwise, (p, q) becomes consecutive. Wlog, suppose e_1 exists. In this case, we replace (p, q) in T by the consecutive arc (p', q'') (see Figure 9(d)). There is no crossings in $T' = (T \setminus \{(p, q)\}) \cup \{(p', q'')\}$, as (p', q'') nests (q', q'') and does not cross any arc in T nested in (p, q). T' is a spanning tree, as all the points are contained in it and the two connected components in $T \setminus \{(p, q)\}$ are joined in T' by the arc (p', q''). As (p', q'') is a consecutive arc nested in the non-consecutive arc (p, q), the cost of T_1 is strictly less than the cost of T. This concludes the proof of the claim and so the proof of the observation. □

As the spanning tree we want to compute is non-crossing, by the above observation, it follows that all the arcs between two consecutive chunks are nested.

Observation 10 *Consider any two arcs (p_1, q_1) and (p_2, q_2) in a minimum non-crossing spanning tree such that $p_1, p_2 \in P_i$ and $q_1, q_2 \in P_{i+1}$. Then, either (p_2, q_2) is nested in (p_1, q_1) or (p_1, q_1) is nested in (p_2, q_2).*

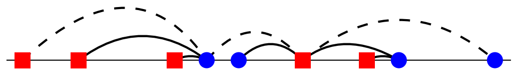


Fig. 10: Figure showing a spanning tree with the umbrella shown by dashed arcs.

Proof. This observation follows from the fact that if none of the two arcs is nested in the other, then they must cross. □

The above observation implies that the outermost arcs between consecutive chunks form a path (an umbrella) between the first and the last point and all the other arcs lie inside this umbrella (see Figure 10). Next, we give a simple algorithm to compute an optimum spanning tree inside such an outermost arc. Suppose $p_0, p_1, \ldots, p_l, \ldots, p_{\tau+1}$ are points in sorted order such that $\{p_0, p_1, \ldots, p_l\} \subseteq P_i$ and $\{p_{l+1}, \ldots, p_{\tau+1}\} \subseteq P_{i+1}$. We would like to construct an optimum spanning tree of the points $p_0, p_1, \ldots, p_l, \ldots, p_{\tau+1}$ which contains the arc $(p_0, p_{\tau+1})$. Our algorithm is based on the following observation.

Observation 11 *Any optimum spanning tree that contains $(p_0, p_{\tau+1})$ must also contain either (p_0, p_τ) or $(p_1, p_{\tau+1})$ whichever has lower weight.*

Proof. Consider any optimum non-crossing spanning tree T that contains the arc $(p_0, p_{\tau+1})$. Also, suppose T does not contain a cheaper arc from $\{(p_0, p_\tau), (p_1, p_{\tau+1})\}$. Otherwise, we are done with the proof. If T does not contain any arc from $\{(p_0, p_\tau), (p_1, p_{\tau+1})\}$, then at least one of p_1 or p_τ cannot be connected to T without introducing any crossings. It follows that T contains the arc e from $\{(p_0, p_\tau), (p_1, p_{\tau+1})\}$ having the larger weight. But, in that case one can replace e in T by the cheaper arc e'. $T' = (T \setminus \{e\}) \cup \{e'\}$ is a non-crossing spanning tree, as the only arc in T that e' crosses is e which is now removed. But, then the weight of T' is strictly lesser than that of T, which is a contradiction. Hence, the observation follows. □

In our algorithm to compute an optimum spanning tree inside an outermost arc $(p_0, p_{\tau+1})$, first we select the shorter arc among (p_0, p_τ) and $(p_1, p_{\tau+1})$. Then, we recursively solve the problem inside the selected arc by treating it as an outermost arc. This problem can be solved in linear time. We are going to use this algorithm as a subroutine in our algorithm for solving the general problem. Next, we give a Dynamic Programming (DP) based algorithm for this general problem. This algorithm essentially decides which outermost arcs to choose.

Let $p_1, p_2, \ldots, p_{n+m}$ be the input points. Our DP algorithm incrementally computes a non-crossing spanning tree starting from the left connecting a new point in each step. Let $P_1 = \{p_1, \ldots, p_l\}$ and $P_2 = \{p_{l+1}, \ldots, p_\tau\}$. For $l+1 \leq i \leq n+m$, let MST(i) be the subproblem of computing an optimum spanning tree for the prefix set of points $\{p_1, \ldots, p_i\}$. We store the cost of MST(i) in table M indexed by i, where $l+1 \leq i \leq n+m$. To initialize, for each $l+1 \leq i \leq \tau$, we compute the cost of an optimum spanning tree of $\{p_1, \ldots, p_i\}$. Note that in this base case, any non-crossing spanning tree contains the outermost arc (p_1, p_i).

Otherwise, either p_1 or p_i cannot be connected without crossing. Thus, we can compute an optimum spanning tree using the algorithm mentioned above for computing an optimum spanning tree inside an outermost arc. Thus, in the base case we compute all the entries of M with indices in $\{l+1, \dots, \tau\}$.

Now, suppose we want to solve MST(i) for any $i \geq \tau+1$. Thus, we would like to connect a new point $p_i \in P_{t+1}$ for some $t \geq 2$. We have already computed the entry $M[q]$ for all $l+1 \leq q < i$. By Observation 9, in the solution spanning tree of MST(i), p_i must be connected to a point p_s of P_t, as $p_i \in P_{t+1}$. For each such p_s, we compute the cost of the spanning tree that contains the arc (p_s, p_i). In particular, the total cost is the sum of three costs: (i) the cost of the arc (p_s, p_i), (ii) the cost of connecting the points inside the open interval (p_s, p_i) and (iii) the cost of the optimum spanning tree of $\{p_1, \dots, p_s\}$. We select the point p_s that minimizes the total cost. Thus,

$$M[i] = \min_{s : p_s \in P_t} \|p_i p_s\| + \mathrm{cost}(s, i) + M[s].$$

Here, $\|p_i p_s\|$ is the cost of (p_s, p_i) and cost(s, i) is the cost of any optimum spanning tree inside the outermost arc (p_s, p_i). Note that one can precompute and store the costs cost(s, i) for all possible open intervals (p_s, p_i) in quadratic time using the algorithm for computing an optimum spanning tree inside an outermost arc. Thus, for a fixed p_s, the sum of the three costs mentioned above can be computed using a constant number of table lookups. Thus, each step of this dynamic programming based algorithm takes linear time. Hence, an optimum spanning tree can be computed in quadratic time.

Theorem 4. *For any set R of red points and B of blue points on a line, an optimum non-crossing spanning tree can be computed in quadratic time.*

4 Minimum Non-crossing Matching for Collinear Points

Note that the fact that a minimum-weight bichromatic matching for points in general position is always non-crossing might not hold in the case of collinear points. Indeed, there are point sets for which no non-crossing matching exists if the edges are represented by segments. However, one can show that there is always a non-crossing matching of collinear points such that each matching edge is a circular arc drawn above the line. Again the weight of an arc is the Euclidean distance between its endpoints. We say that two arcs are *pairwise disjoint* if their endpoints are disjoint.

Definition 4. ***Non-crossing matching for collinear points***. *Given a set R of n red points and a set B of n blue points all of which lie on a line, find a set of n pairwise disjoint and non-crossing circular arcs in the plane of minimum total weight such that the arcs lie above H, each arc connects a red and a blue point, and the arcs span all the input points.*

Using the bipartite matching algorithm due to Kaplan *et al.* [19] along with a simple postprocessing (already described in the introduction), one can immediately solve this problem in $O(n^2 \operatorname{poly}(\log n))$ time. Here we design a simple algorithm with improved $O(n)$ time complexity.

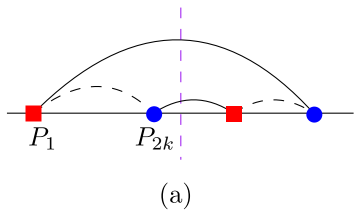
Let $p_1, p_2, \ldots, p_{2n}$ be the input points sorted from left to right based on their x coordinates. We assume that the points are given in this order. For any point $p_i \in P$, let $\operatorname{col}(p_i)$ denote the color of p_i. A subset of points $P_i \subseteq P$ is called *color-balanced* if it contains an equal number of red and blue points. We traverse the points from left to right and seek the first balanced subset (denoted by P_1). In order to obtain P_1 we use a simple method. We start with the leftmost point p_1 and maintain a counter C which is used to find the balanced subset and is initialized to 0 at the beginning. If $\operatorname{col}(p_1) = \text{red}$, we increase the value of C by 1, and decrease by 1, otherwise. Observe that we will get a balanced subset when the value of C becomes 0. Let $P_1 \subseteq P$ be the first balanced subset containing $2k$ (for some $k \in \{1, \ldots, n\}$) points. The remaining points $P \setminus P_1$ also form a balanced subset since P contains exactly n red and n blue points. We prove the following lemma.

Lemma 12. *Let $P_1 \subseteq P$ be the first color-balanced subset of P and $|P_1| = 2k$. Then $\operatorname{col}(p_1) \neq \operatorname{col}(p_{2k})$, and any minimum non-crossing perfect matching M_P of P contains the edge (p_1, p_{2k}).*

Proof. The first part of the lemma is clearly true, otherwise the value of the counter would not be 0 at p_{2k}, which is the termination criteria to obtain the first balanced subset. Now, let us assume that M_P does not contain the edge (p_1, p_{2k}). Then one of the following two situations can happen: 1) p_1 and p_{2k} are matched with two intermediate points from P_1; 2) one or both of p_1 and p_{2k} are matched with points from $P \setminus P_1$.

Case 1: p_1 and p_{2k} are matched with two intermediate points p_τ and p_ℓ, respectively. Note $\ell > \tau$, otherwise the matching edges cross each other. We know that $\{p_1, \ldots, p_\tau\}$ is not a balanced subset since P_1 is the first balanced subset. Therefore, there exists at least one point p_r (where $1 < r < \tau$) that is matched with a point p_s (where $s > \tau$). In that case, the edge (p_r, p_s) will intersect (p_1, p_τ). Hence, we get a contradiction.

Case 2: Suppose both of p_1 and p_{2k} are matched with points from $P \setminus P_1$ and no other point from $\{p_2, \ldots, p_{2k-1}\}$ is matched with any point from $P \setminus P_1$. Then we can construct a new matching by adding the edge (p_1, p_{2k}) and by matching the two points in $P \setminus P_1$. The new matching has lesser cost and is non-crossing; see Figure 11(a). If any other point in $\{p_2, \ldots, p_{2k-1}\}$ (say p_x) is also matched with a point in $P \setminus P_1$, then we know it must be of opposite color of either p_1 or p_{2k}, since $\operatorname{col}(p_1) \neq \operatorname{col}(p_{2k})$. Hence, we can either add the edge (p_1, p_x) or (p_x, p_{2k}) and this reduces the total cost; see Figure 11(b). The new matching might not be non-crossing. But, using similar argument one can remove all the crossings without increasing the cost. Thus, at the end we get a cheaper non-crossing matching, which contradicts the optimality of M_P.



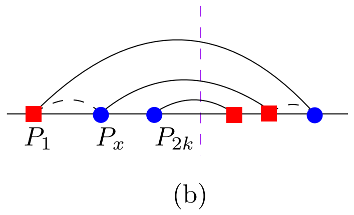


Fig. 11: Figure demonstrating the two situations in the case when both p_1 and p_{2k} are matched with points from $P \setminus P_1$.

Now, if only one of p_1 or p_{2k} is matched with a point from $P \setminus P_1$, let us assume p_1, then we know there must be at least one other point (say $p_x \in P_1$) that is also matched with a point from $P \setminus P_1$, and $\text{col}(p_1) \neq \text{col}(p_x)$. We can apply similar arguments as above to get a contradiction, which concludes the proof of the lemma. □

Now, we use Lemma 12 to proceed with the algorithm. First, we obtain the balanced subset P_1, and match the points p_1 and p_{2k} by an arc and include the edge (p_1, p_{2k}) in M_P. This edge partitions the point set into two color-balanced subsets, i.e., $P_2 = P \setminus P_1$ and $P_1' = P_1 \setminus \{p_1, p_{2k}\}$. On each of these subsets we recursively perform the same procedure. This process is repeated until each point of P is matched.

Due to Lemma 12, we know that every edge we choose in our algorithm must be part of the optimum solution, and no two edges cross each other. Next we show how to convert our recursive algorithm to a non-recursive one, in order to implement it in linear time. Consider the points in left to right order, and insert the leftmost point p_1 onto a stack. Now, if the next point p_2 is of same color as the stack top p_1, then push p_2 onto the stack; otherwise, match p_1, p_2 and remove p_1 from the stack. Repeat this process until all points are considered. Indeed, this algorithm is same as the algorithm for matching of parentheses.

The above linear time algorithm has the same effect as the recursive algorithm. The only difference is that like before when p_1 is matched with p_{2k}, now all the points in between p_1 and p_{2k} are already matched. This can be proved by induction, as $P_1' = P_1 \setminus \{p_1, p_{2k}\}$ is also a color-balanced set. We conclude with the following theorem.

Theorem 5. *For any set R of red points and B of blue points on a line with $|R| = |B|$, an optimum non-crossing matching can be computed in linear time.*

5 Conclusion and Open Problems

In this paper, we have studied three classical graph problems on geometric graphs induced by bichromatic points for collinear points. We have shown that almost all of these problems can be solved in linear time in this setting. We note that the results for circular-arc edges trivially extend to other types of edges drawn in a topologically equivalent way within the same halfplane, e.g, 1-bend polylines. One problem that is left open by our work is the complexity of Bichromatic TSP for collinear points. We have shown that if a Hamiltonian path is forced to lie above the line, then such a path might not exist. One interesting question is to design an algorithm for finding a path in this setting if one exists. Another interesting open problem is to design a subquadratic time algorithm for computing a non-crossing spanning tree.

References

1. A. K. Abu-Affash, A. Biniaz, P. Carmi, A. Maheshwari, and M. H. M. Smid. Approximating the bottleneck plane perfect matching of a point set. *Comput. Geom.*, 48(9):718–731, 2015.
2. P. K. Agarwal, A. Efrat, and M. Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIAM J. Comput.*, 29(3):912–953, 1999.
3. O. Aichholzer, C. Alegría, I. Parada, A. Pilz, J. Tejel, C. D. Tóth, J. Urrutia, and B. Vogtenhuber. Hamiltonian meander paths and cycles on bichromatic point sets. In *XVIII Spanish Meeting on Computational Geometry*, pages 35–38, 2019.
4. S. Anily and R. Hassin. The swapping problem. *Networks*, 22(4):419–433, 1992.
5. S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998.
6. F. Bernhart and P. C. Kainen. The book thickness of a graph. *J. Combinatorial Theory, Series B*, 27(3):320–331, 1979.
7. A. Biniaz, P. Bose, D. Eppstein, A. Maheshwari, P. Morin, and M. Smid. Spanning trees in multipartite geometric graphs. *Algorithmica*, 80(11):3177–3191, 2018.
8. A. Biniaz, P. Bose, A. Maheshwari, and M. H. M. Smid. Plane geodesic spanning trees, Hamiltonian cycles, and perfect matchings in a simple polygon. *Comput. Geom.*, 57:27–39, 2016.
9. A. Biniaz, P. Bose, A. Maheshwari, and M. H. M. Smid. Plane bichromatic trees of low degree. *Discrete & Computational Geometry*, 59(4):864–885, 2018.
10. M. G. Borgelt, M. J. van Kreveld, M. Löffler, J. Luo, D. Merrick, R. I. Silveira, and M. Vahedi. Planar bichromatic minimum spanning trees. *J. Discrete Algorithms*, 7(4):469–478, 2009.
11. J. Colannino, M. Damian, F. Hurtado, S. Langerman, H. Meijer, S. Ramaswami, D. L. Souvaine, and G. Toussaint. Efficient many-to-many point matching in one dimension. *Graphs Comb.*, 23(Supplement-1):169–178, 2007.
12. S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani. *Algorithms.* McGraw-Hill, 2008.
13. W. S. Evans, G. Liotta, H. Meijer, and S. K. Wismath. Alternating paths and cycles of minimum length. *Comput. Geom.*, 58:124–135, 2016.
14. A. Frank, E. Triesch, B. Korte, and J. Vygen. On the bipartite travelling salesman problem. 1998.
15. A. García and J. Tejel. Polynomially solvable cases of the bipartite traveling salesman problem. *European Journal of Operational Research*, 257(2):429–438, 2017.
16. A. Kaneko and M. Kano. Straight-line embeddings of two rooted trees in the plane. *Discrete & Computational Geometry*, 21(4):603–613, 1999.

17. A. Kaneko and M. Kano. Discrete geometry on red and blue points in the plane—a survey—. In *Discrete and computational geometry*, pages 551–570. Springer, 2003.
18. A. Kaneko, M. Kano, and K. Suzuki. Balanced partitions and path covering of two sets of points in the plane. *preprint*.
19. H. Kaplan, W. Mulzer, L. Roditty, P. Seiferth, and M. Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. In *Symposium on Discrete Algorithms, SODA 2017*, pages 2495–2504, 2017.
20. J. Kyncl, J. Pach, and G. Tóth. Long alternating paths in bicolored point sets. *Discrete Mathematics*, 308(19):4315–4321, 2008.
21. W. Mulzer and P. Valtr. Long alternating paths exist. In S. Cabello and D. Z. Chen, editors, *36th International Symposium on Computational Geometry, SoCG 2020, June 23-26, 2020, Zürich, Switzerland*, volume 164 of *LIPIcs*, pages 57:1–57:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.