Geometric Planar Networks
on Bichromatic Points[*]

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

[1] Department of Informatics, University of Bergen, Norway
{sayan.bandyapadhyay}@gmail.com
[2] School of Computer Sciences, NISER, Bhubaneswar, India
{aritrabanik}@gmail.com
[3] Algorithms and Complexity Group, Technische Universität Wien, Austria
{sujoy,noellenburg}@ac.tuwien.ac.at

Abstract. We study four classical graph problems – Hamiltonian path, Traveling salesman, 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 in two restricted settings: (i) collinear points and (ii) equidistant points on a circle. We show that almost all of these problems can be solved in linear time in these constrained, yet non-trivial settings.

1   Introduction

In this article, we study four 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 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 of $G(R, B, E)$ (or equivalently a subset of edges of E) is called non-crossing (or planar) if no pair of the edges of the subgraph cross each other. Next, we discuss the four graph problems on the bipartite graph $G(R, B, E)$ induced by $R \cup B$.

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
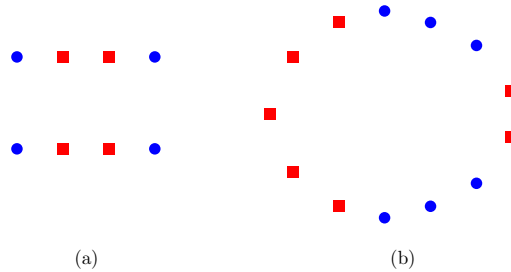
Fig. 1: Two instances without a non-crossing Hamiltonian path. Figure (a) is not in general position, Figure (b) (borrowed from [12]) is in general position.

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 $m - 1 \leq n \leq m + 1$, where it is not possible to find any non-crossing Hamiltonian path. In Figure 1, we demonstrate two such instances.[4] 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 pair of the points are collinear, and 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 [13]. 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 [11,15].

A related problem is the BICHROMATIC TRAVELING SALESMAN PATH (BICHRO-MATIC TSP) problem where one would like to find a minimum weight Hamiltonian path in $G(R, B, E)$. The 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. For simplicity, we assume $n = m$. A straightforward reduction from the (monochromatic) Euclidean TSP [4] (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 [4] 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. [10] who improved the 2.5-approximation of Anily et al. [3]. For a set of collinear points, Evans et al. [9] gave a quadratic time algorithm for computing an op-

---

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

timum non-crossing TSP, every edge of which is a poly-line with at most two bends.

Next, we consider the BICHROMATIC SPANNING TREE problem where the objective is to compute a minimum weight spanning tree of $G(R, B, E)$. Note that this problem can be solved efficiently by any standard minimum spanning tree algorithm. A more interesting problem is the NON-CROSSING BICHROMATIC SPANNING TREE problem where additionally the computed tree must be planar (non-crossing). Borgelt *et al.* [8] 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 [7].

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 Bichromatic matching exactly. But, algorithms with better running time have been designed by exploiting the underlying geometry of the plane. Recently, Kaplan *et al.* [14] 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. Abu-Affash *et al.* [1] and Biniaz *et al.* [6] studied variants of the bichromatic matching problem.

In this article, we consider the above mentioned problems in two restricted settings: (i) for collinear points and (ii) for equidistant points on a circle. For all problems, we assume that $n = m$ for simplicity. We note that the case of non-crossing graphs on collinear points is closely related to topological 1-page or 2-page book embeddings [5], which have all vertices placed on a line (called the spine) and the edges drawn without crossings in one or two of 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.[5] 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.

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, there always exists a non-crossing Hamiltonian path. We give a linear-time algorithm for computing such a path (Section 2).

---

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

➥ 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).

➥ 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).

➥ TS TOUR FOR CHUNKED POINTS ON A CIRCLE – We give a linear-time algorithm for computing an optimum traveling salesman tour for equidistant points on a circle that form alternately colored and equally sized chunks (Section 5).

The linear-time algorithms assume that the points are given in some sorted order. 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. Throughout the paper we assume that there are no collocated points.

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 of n red points and a set of n blue points on the line $H : y = 0$, find a non-crossing geometric path π in the plane such that π consists 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 (see Figure 2).

First, we give a constructive proof for the existence 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. In the following, we describe the construction.
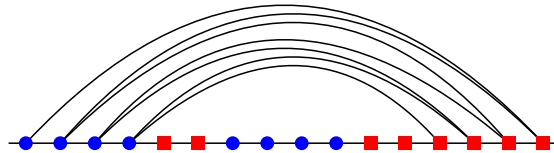

Fig. 2: Figure demonstrating a set of collinear points for which non-crossing Hamiltonian path does not exist if the arcs can be drawn only above the line.

2.1 The Construction

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

Definition 2. *A set of matched edges M are hierarchical 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 uncrossed) or $u < w < x < v$ ((u, v) contains (w, x)).*

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 disjoint and none of them contains 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 $(u, w), (v, x)$. Note that in all the cases, the new pair of edges do not violate the hierarchical structure. We repeat the process for each pair of edges that violate the condition. Newly formed edges might violate the condition with respect to other edges. However, it is easy to verify that if an edge is removed, it is never added back, and thus the process will eventually stop at some point when no pair of edges violate the condition.

Next, we associate levels with each matched edge of M in a recursive way. In the base case, for each edge that does not span any other edge, set its level to 1. Now, suppose we have defined edges of level j for each $j \le i - 1$ for $i \ge 2$. An edge (u, v) has level i, if it contains a level $i - 1$ edge, and for any level $i - 1$ edge (w, x) that it contains, there is no other edge that is contained in (u, v) that also contains (w, x) (see Figure 3). Note that the level of each edge is unique. Let L be the maximum level.
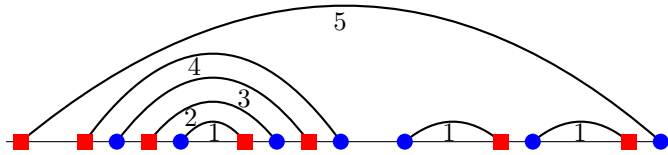

Fig. 3: Illustration showing the levels of the edges.

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 blocks of levels at most $l - 1$ and two special points which are the first and last point of the block. One can easily verify that a block contains the same number of

red and blue points. We compute the Hamiltonian path for all the blocks in a bottom up manner. The path of a level 1 block is the matched 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 $j \le l - 1$ and $l \ge 2$ 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 $2 \le i \le t$. 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 1. *A non-crossing Hamiltonian path of $\cup_{j=1}^{i} S_j$ can be computed for all $1 \le i \le t$ that satisfies the two invariants.*
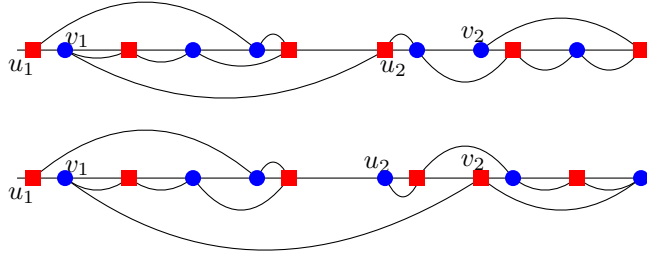

Fig. 4: Illustration showing Case 1 (upper) and Case 2 (lower) of Lemma 1.

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 $i \ge 2$. 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.
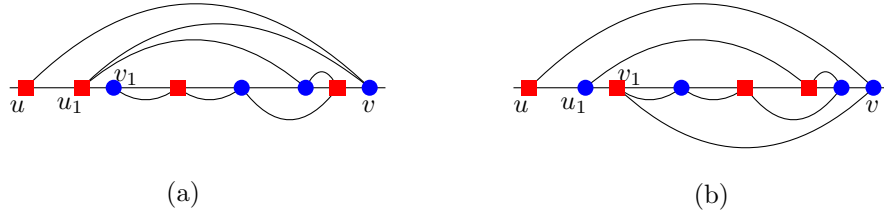
Fig. 5: (a) Illustration showing Case 1 of Lemma 2. (b) Illustration showing Case 2 of Lemma 2.

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 4). 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 4). Note that 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. Note that 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 2. *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 1. Let u_1, v_1 be the endpoints of T_t such that $u_1 < v_1$. Note that as mentioned before $S = (\cup_{j=1}^t S_j) \cup \{u, v\}$. WLOG, 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 5(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 5(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 a union of a set of blocks having levels at most the maximum level L. By Lemma 2, we can compute the paths for all such blocks that satisfy the invariants. Then we can merge those paths using the construction in Lemma 1 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 $y = 0$ with $|R| = |B|$, there always exists a non-crossing Hamiltonian path whose edges are circular arcs that lie above or below $y = 0$. Moreover, such a path can be computed in polynomial time.*

2.2   A Linear Time Algorithm for Non-Crossing Hamiltonian Path

Recall that all the input points lie on $H : y = 0$. We assume that the points are given in sorted order w.r.t 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. The 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, and all points are active. We store the constructed path in a set of edges Π, which is initially empty.

- Let $l(r)$ and $l(b)$ 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 a small arc $(p, S(p))$ to Π that lie above H. Make p inactive.
- Otherwise, there are two cases.
    (i). If p is red, add two edges $(p, l(b))$ and $(l(b), S(p))$ to Π. These two edges are drawn above H as circular arcs. Make p and $l(b)$ inactive.

(ii). If p is blue, add two edges $(p, l(r))$ and $(l(r), S(p))$ to Π. These two edges are drawn below H as circular arcs. Make p and $l(r)$ 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 6. Now we discuss the correctness of the algorithm. First, we have the following observation.
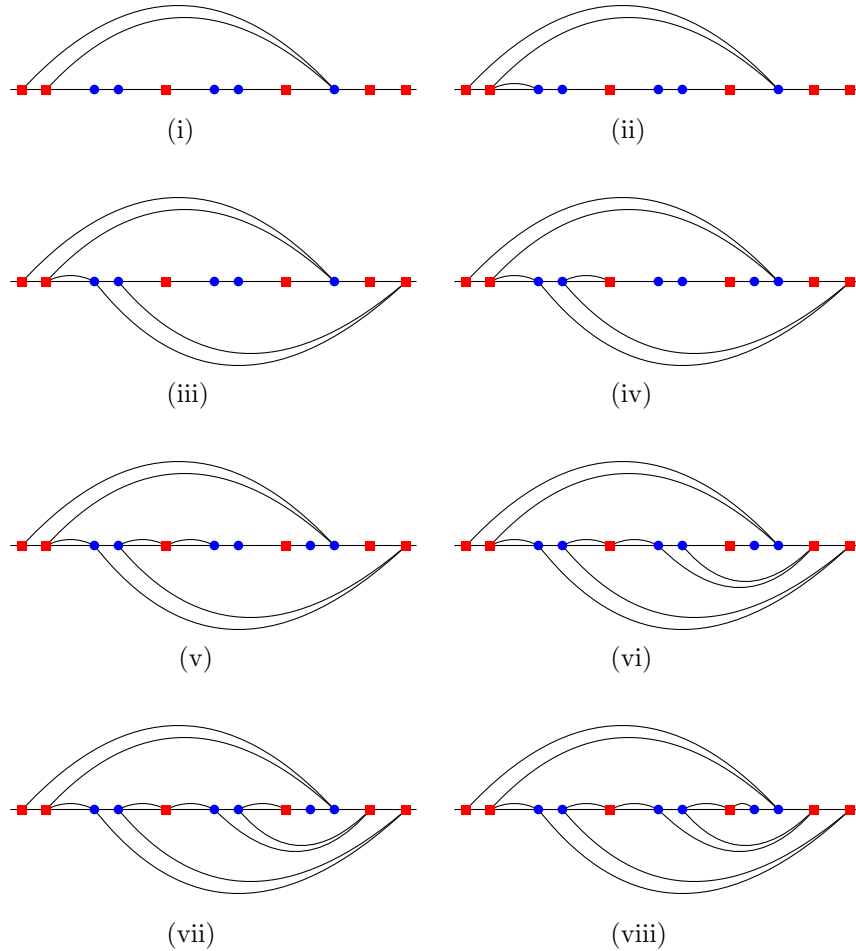

Fig. 6: Figure showing the execution of the Hamiltonian path computation algorithm on an example point set.

Observation 3 *Consider any iteration of the algorithm. Then, any red point on the right of $l(r)$ (if any) is inactive and has degree 2. Similarly, any blue point*

on the right of $l(b)$ (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 4. *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 $l(r)$ (resp. $l(b)$) at some point and its degree is 2. By Observation 3, 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. It is easy to see that 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 valid bichromatic Hamiltonian path.   □

Next, we argue that the computed Hamiltonian path is non-crossing. It is easy to see that the small arcs added in the second step do not cross any other drawn edges. Also, the edges drawn above H do not cross any edges drawn below H. Moreover, the edges $(p, l(r))$ and $(l(r), S(p))$ (or $(p, l(b))$ and $(l(b), S(p))$) drawn in the same iteration do not cross each other. The following observation completes the claim.

Observation 5 *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 lie in between u' and v', or u', v' lie in between u and v.*

The algorithm can be implemented to run in linear time. This is because, one can use three pointers to keep track of p, $l(r)$ and $l(b)$, and these pointers move in one direction – either from left to right or from right to left.

Theorem 2. *For any set R of red points and B of blue points on $y = 0$ with $|R| = |B|$, a non-crossing Hamiltonian path can be computed in linear time whose edges are circular arcs that lie above or below $y = 0$.*

3   Minimum Spanning Tree for Collinear Points

Definition 3. *Spanning tree for collinear points. Given a set of n red points and a set of n blue points all of which lie on the line $y = 0$, 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 $y = 0$, 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 Crossing

Let $p_1, \ldots, p_{2n}$ be the input points sorted in increasing order of their values. 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. This gives us a set of components $\{C_1, \ldots, C_m\}$ for $m \in [n]$, where each component contains at least one edge. For any component $C_i, \forall i \in [m]$, 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 $col(r(C_1)) \neq col(l(C_1))$, 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 one to join C_1, and C_2. We repeat the same process for each consecutive pair of the remaining components.
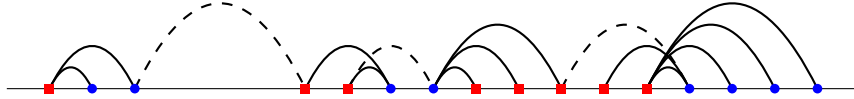

Fig. 7: Figure demonstrating the execution of the algorithm on an example. The dashed arcs are added in the second step.

Note that after the first step each component C_i is a tree. In the second step, we add exactly one edge between a consecutive pair of components. Hence, the selected arcs form a valid spanning tree. The optimality of the solution follows from the next observation which is not hard to verify.

Observation 6 *Cut property: Any edge (or arc) added by the algorithm is a minimum weight bichromatic cut edge for some cut.*

We note that our algorithm does not require the assumption $|R| = |B|$.

Theorem 3. *For any set R of red points and B of blue points on $y = 0$, an optimum spanning tree can be computed in linear time.*

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

3.2 Non-crossing Spanning Tree

Let $P_1, P_2, \ldots, P_m$ be the alternating monochromatic chunks of points ordered from left to right for $m \in [n]$. Thus, the color of the points in P_i is different from the color of the points in P_{i+1} for all $1 \leq i \leq m-1$. We start with the following observation.

Observation 7 *Consider a 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 $q \in P_{i+1}$.*

The observation follows from the idea that if a point is connected to a point that belongs to a non-consecutive chunk, then one can find a cheaper spanning tree by replacing the connecting arc with another arc having lower weight. 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 8 *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 $p_1 < p_2 < q_2 < q_1$ or $p_2 < p_1 < q_1 < q_2$.*

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 8). Next, we give a simple algorithm to compute an optimum spanning tree inside such an outermost arc. Suppose $p_0, p_1, \ldots, p_l, \ldots, p_{k+1}$ be points in sorted order such that $\{p_0, p_1, \ldots, p_l\} \subseteq P_i$ and $\{p_{l+1}, \ldots, p_{k+1}\} \subseteq P_{i+1}$. We would like to construct an optimum spanning tree of the points $p_0, p_1, \ldots, p_l, \ldots, p_{k+1}$ which contains the arc (p_0, p_{k+1}). Our algorithm is based on the following observation.
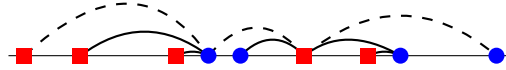

Fig. 8: Figure showing a spanning tree with the umbrella shown by dashed arcs.

Observation 9 *Any optimum spanning tree that contains (p_0, p_{k+1}) must also contain either (p_0, p_k) or (p_1, p_{k+1}) whichever has lower weight.*

In our algorithm, we select the shorter arc among (p_0, p_k) and (p_1, p_{k+1}). Then, we recursively solve the problem inside the selected arc by treating it as an outermost arc. It is easy to see that this problem can be solved in linear time. Next, we give an algorithm for deciding which outermost arcs to choose.

Let $p_1, p_2, \ldots$ be the input points. Our algorithm incrementally computes a non-crossing spanning tree starting from the left and by connecting a new point in each step. Let $P_1 = \{p_1, \ldots, p_l\}$ and $P_2 = \{p_{l+1}, \ldots, p_k\}$. To initialize, for each $l+1 \leq j \leq k$, we compute the cost of optimum spanning tree of $\{p_1, \ldots, p_j\}$ that contains the outermost arc (p_1, p_j) using the above algorithm. Now, suppose we want to connect a new point $p_i \in P_{t+1}$ for $t \geq 2$. We have already computed the cost of an optimum spanning tree of $\{p_1, \ldots, p_q\}$ with any valid outermost arc (r, s), where $r \in P_{t-1}$ and $s \in P_t$. In the new spanning tree, p_i must be connected to a point s of P_t. For each such s, we compute the cost of the spanning tree that contains the arc (s, p_i). In particular, the total cost is the sum of three costs: (i) the cost of (s, p_i), (ii) the cost of connecting the points inside (s, p_i) and (iii) the cost of the optimum spanning tree of $p_1, \ldots, s$ that contains (r, s) for some $r \in P_{t-1}$. We select the arc (r, s) in our spanning tree that minimizes the total cost.

Note that each step of this dynamic programming based algorithm takes linear time. Thus, the optimum spanning tree can be computed in quadratic time.

Theorem 4. *For any set R of red points and B of blue points on $y = 0$, 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 matched edge is a circular arc drawn above the line. Again the weight of an arc is the Euclidean distance between its endpoints.

Definition 4. *Non-crossing matching for collinear points. Given a set of n red points and a set of n blue points all of which lie on the line $y = 0$, find a set of n non-crossing circular arcs in the plane of minimum total weight such that the arcs lie above $y = 0$, 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.* [14] 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 $col(p_i)$ 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 for 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 $col(p_1) = $ 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 $2m$ (for some $m \in [n]$) 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 10. *Let $P_1 \subseteq P$ be the first color-balanced subset of P and $|P_1| = 2m$. Then $col(p_1) \neq col(p_{2m})$, and any minimum non-crossing perfect matching M_P of P contains the edge (p_1, p_{2m}).*

Proof. The first part of the lemma is clearly true, otherwise the value of the counter would not be 0 at p_{2m}, which is the termination criteria to obtain the
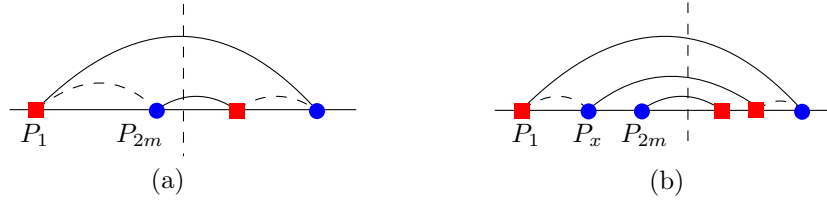
Fig. 9: Figure demonstrating the two situations in the case when both p_1 and p_{2m} are matched with points from $P \setminus P_1$.

first balanced subset. Now, let us assume that M_P does not contain the edge (p_1, p_{2m}). Then one of the following two situations can happen: 1) p_1 and p_{2m} are matched with two intermediate points from P_1; 2) one or both of p_1 and p_{2m} are matched with points from $P \setminus P_1$.

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

Case 2: Suppose both of p_1 and p_{2m} are matched with points from $P \setminus P_1$ and no other point from $\{p_2, \ldots, p_{2m-1}\}$ is matched with any point from $P \setminus P_1$. Then we can construct a new matching by adding the edge (p_1, p_{2m}) and by matching the two points in $P \setminus P_1$. The new matching has lesser cost and is non-crossing; see Figure 9(a). If any other point in $\{p_2, \ldots, p_{2m-1}\}$ (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_{2m}, since $col(p_1) \neq col(p_{2m})$. Hence, we can either give the edge (p_1, p_x) or (p_x, p_{2m}) and this reduces the total cost; see Figure 9 (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.

Now, if only one of p_1 or p_{2m} is matched (WLOG, assume it is p_1) with a point from $P \setminus P_1$, 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 $col(p_1) \neq col(p_x)$. We can apply similar arguments as above to get a contradiction, which concludes the proof of the lemma. □

Now, we use Lemma 10 to proceed with the algorithm. First, we obtain the balanced subset P_1, and match the points p_1 and p_{2m} by an arc and include the edge (p_1, p_{2m}) in M_P. This edge partitions the point set into subsets, i.e., $P_2 = P \setminus P_1$ and $P_1' = P_1 \setminus \{p_1, p_{2m}\}$. 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 10, we know that every edge we choose in our algorithm must be part of the optimum solution, and no two edges cross each other. It is not hard to see that all the balanced subsets can be computed in linear time in advance, as they are corresponding to matched parentheses and are at most n in number[6]. Thus, we conclude with the following theorem.

Theorem 5. *For any set R of red points and B of blue points on $y = 0$ with $|R| = |B|$, an optimum non-crossing matching can be computed in linear time.*

5   TS Tour for Chunked Points on a Circle

Finally, we study the Traveling Salesman problem on the following special point configuration on a circle.

Definition 5. *TS tour for chunked points on a circle. We are given a set of n red points and a set of n blue points all of which lie on a fixed circle. All points are distributed equidistantly on the circle. Further, the input points are divided into alternately-colored chunks, where each chunk contains exactly k consecutive points of the same color. The goal is to find a geometric (closed) tour π in the plane of minimum total length such that π consists of segments each of which connects a red and a blue point and π spans all the input points.*
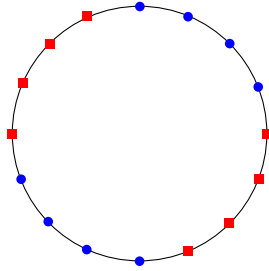

Fig. 10: Figure demonstrating the point configuration we consider for TS tour, where $n = 8$ and $k = 4$.

Note that by definition, n/k is an integer. The total number of chunks is $L = 2n/k$ of which n/k contain only red points and n/k contain only blue points. The configuration of the points mentioned in the above definition is shown in Figure 10 with an example. For any arc between two points u and v

---

[6] This algorithm can be easily implemented in the following manner. Consider the points in left to right order, and insert the leftmost point (p_1) into a stack. Now, if the next point p_2 is of same color as p_1 then insert p_2 into the stack, otherwise match p_1, p_2 and remove p_1 from the stack. Repeat this process until all points are considered.

on the circle C, we denote the arc by $c(uv)$ (resp. $a(uv)$) if it is the clock (resp. anticlock) -wise traversal from u to v along C. As any two consecutive points are a fixed distance apart we measure the length of any bichromatic edge (a straight line segment) uv by the minimum of the number of points on the arcs $c(uv)$ and $a(uv)$, respectively (including u and v). Next, we design an algorithm for computing a TS tour for the input points. We consider two cases: (i) k is even and (ii) k is odd.

(i) k **is Even**. The algorithm in this case is as follows.

1. Let $2p = k$. Partition each chunk into two subchunks each containing p consecutive points. Merge all consecutive subchunks of different colors to form groups. Note that each group contains p red and p blue points. We still preserve the geometry of the points of each group and identify the two peripheral red and blue points of each group as special points. We first compute a bichromatic path between the two special points for each group and later connect the special points of different groups to construct a TS tour for all the points.

2. For each group, we compute the TS tour in the following way. Consider the ordering of the groups w.r.t clockwise traversal of the points and consider the ith group in this order. WLOG, assume that the red points are visited before the blue points while traversing the points of the group in clockwise order. Let $r_1^i, r_2^i, \ldots, r_p^i, b_p^i, b_{p-1}^i, \ldots, b_1^i$ be the points in this order. Join r_p^i with b_p^i and b_{p-1}^i using two edges. For each $p-1 \geq j \geq 2$, join r_j^i with b_{j+1}^i and b_{j-1}^i. Finally, join r_1^i with b_2^i (see Figure 11). Note that each of the points in the group except r_1^i and b_1^i is connected to two points. r_1^i and b_1^i are connected to only one point.

3. Next, we connect the special points of different groups. Recall that L is the total number of groups. Let for the first group the red points are visited before the blue points while traversing the points of the group in clockwise order. Note that the special points are $r_1^1, b_1^1, b_1^2, r_1^2, r_1^3, b_1^3, \ldots, b_1^L, r_1^L$. For $1 \leq i \leq L-1$, we connect r_1^i to b_1^{i+1}. We also connect r_1^L to b_1^1 (see Figure 11).

It is not hard to see that the set of selected edges form a valid traveling salesman tour. This is because each of the points is connected to exactly two other points of opposite color. Now, we give a bound on the length of the tour.

Lemma 11. *The length of the computed tour is $n(k + 2 + 2/k)$.*

Proof. Note that we have added two types of edges (1) between points of same group, and (2) between the special points. The length of an edge of the second type is exactly $k + 1$, and the number of those edges is L. For each group, we have $k - 1$ type 1 edges whose total length is,

$$2 + (3 + 3) + (5 + 5) + \ldots + (k - 1 + k - 1)$$
$$= 2 \cdot (1 + 3 + \ldots + k - 1)$$
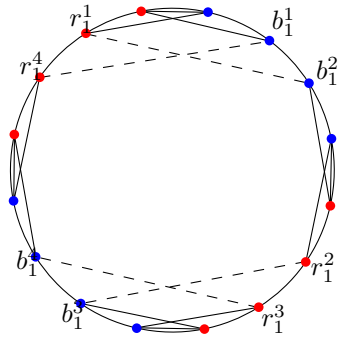$$= 2 \cdot (k^2/4) = k^2/2$$

Fig. 11: An example of the tour computation for the even case. The edges between points of same (resp. different) groups are shown using regular (resp. dashed) segments.

Thus, for all the groups, the total length of the type 1 edges is $Lk^2/2$. Hence, the total length of all edges is $L((k^2 + 2k + 2)/2) = n(k + 2 + 2/k)$. $\square$

The algorithm for the odd case is as follows.

1. Let $2p + 1 = k$. Partition each chunk without the middle point into two subchunks each containing $p - 1$ consecutive points. Add the middle point to both subchunks. Merge all consecutive subchunks of different colors to form groups. Note that each group contains p red and p blue points. We still preserve the geometry of the points of each group and identify the two peripheral red and blue points as special points. For each group, We compute a bichromatic path between the two special points. As each pair of consecutive groups share a special point, the union of the paths corresponding to the groups form a valid tour.
2. For each group, we compute the TS tour exactly in the same way as in the k is even case.
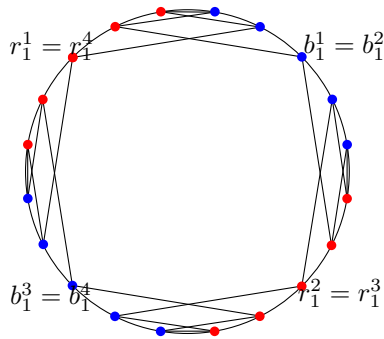

Fig. 12: An example of the tour computation for the odd case.

Now, we give a bound on the length of the tour.

Lemma 12. *The length of the computed tour is $n(k + 2 + 1/k)$.*

Proof. For each group, we have k edges whose total length is,

$$2 + (3 + 3) + (5 + 5) + \ldots + (k + k)$$
$$= 2 \cdot (1 + 3 + \ldots + k)$$
$$= (k + 1)^2/2$$

Thus, for all the groups, the total length of the edges is $(2n/k) \cdot ((k^2 + 2k + 1)/2) = n(k + 2 + 1/k)$. $\square$

Note that the algorithm can be easily implemented in linear time assuming the points are given in clockwise or anticlockwise order.

5.1 Lower Bound

Lemma 13. *The length of any bichromatic traveling salesman tour for the configuration of the points on the circle is at least $n(k + 2 + 2/k)$ if k is even and at least $n(k + 2 + 1/k)$ if k is odd.*

Proofsketch. Assume k is even. The odd case can be handled in a similar way. We will show that the weight of the edges adjacent to each red chunk is at least $k^2 + 2k + 2$ in any tour. As the edges adjacent to two distinct red chunks are disjoint, and the number of red chunks is n/k the lemma follows. Consider a fixed red chunk. The k red points in the chunk must be connected to blue points with $2k$ distinct edges. WLOG, we assume that the only blue points with which these k points are connected are the $2k$ blue points of the two adjacent chunks. Partition the red chunk into two subchunks each containing $k/2$ consecutive points. Again WLOG, we can assume that the points of a subchunk are only connected to the blue points of the adjacent chunk. Now, consider a fixed subchunk. Note that the $k/2$ points should have a total of degree k which will come from the k blue points. Also each blue point can be connected to only two red points. If the degree k come from only $k/2$ blue points, then the subtour involving the $k/2$ red and $k/2$ blue points form a cycle. Hence, the $k/2$ red points connect to at least $k/2 + 1$ blue points. Also it is beneficial to connect the red points to the closest $k/2 + 1$ blue points among the points of the blue chunk. Moreover, to satisfy a total degree k from these $k/2 + 1$ blue points, it is always beneficial to have only one degree from the farthest and the second farthest blue points, and two degree from each of the remaining $k/2 - 1$ points.

Now, there cannot be the case that the first red point r_1 (adjacent to the blue chunk) gets connected to the farthest and the second farthest blue points. Otherwise, the remaining $k/2 - 1$ red and $k/2 - 1$ blue points will form a cycle. Thus, at most one among the farthest and the second farthest blue points gets connected to r_1. The cost to connect the farthest blue point is at least $k/2 + 2$. The edges that connect the remaining $k/2$ blue points and $k/2$ red points form a

path between the second farthest blue point and a red point. The idea is to prove that the cost to connect these points is at least $(k^2 + k - 2)/2$. Thus, the total cost is at least $(k^2 + 2k + 2)/2$ for one red subchunk. For both red subchunks the cost is $k^2 + 2k + 2$, and thus the lemma follows.   □

Theorem 6. *For any set R of red points and B of blue points on a circle with $|R| = |B|$, an optimum non-crossing TS tour can be computed in linear time.*

6   Conclusion and Open Problems

In this paper, we have studied four classical graph problems on geometric graphs induced by bichromatic points in two restricted settings: (i) collinear points and (ii) equidistant points on a circle. We have shown that almost all of these problems can be solved in linear time in these settings. For the case of collinear points, the results are obtained for graphs whose edges can be drawn as circular arcs. We note that the results for collinear points and circular-arc edges trivially extend to other types of edges drawn topologically equivalent within the same halfplane, e.g, 1-bend polylines. One problem that is left open by our work is to decide the complexity of Bichromatic TSP for collinear points. Also, it would be nice to extend our result for chunked points to any configuration of points on a circle.

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. S. Anily and R. Hassin. The swapping problem. *Networks*, 22(4):419–433, 1992.
4. S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998.
5. F. Bernhart and P. C. Kainen. The book thickness of a graph. *J. Combinatorial Theory, Series B*, 27(3):320–331, 1979.
6. 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.
7. 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.
8. 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.
9. W. S. Evans, G. Liotta, H. Meijer, and S. K. Wismath. Alternating paths and cycles of minimum length. *Comput. Geom.*, 58:124–135, 2016.
10. A. Frank, E. Triesch, B. Korte, and J. Vygen. On the bipartite travelling salesman problem. 1998.

11. A. Kaneko and M. Kano. Straight-line embeddings of two rooted trees in the plane. *Discrete & Computational Geometry*, 21(4):603–613, 1999.
12. A. Kaneko and M. Kano. Discrete geometry on red and blue points in the planea survey. In *Discrete and computational geometry*, pages 551–570. Springer, 2003.
13. A. Kaneko, M. Kano, and K. Suzuki. Balanced partitions and path covering of two sets of points in the plane. *preprint*.
14. 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.
15. J. Kyncl, J. Pach, and G. Tóth. Long alternating paths in bicolored point sets. *Discrete Mathematics*, 308(19):4315–4321, 2008.