Faster Algorithms for the Minimum Red-Blue-Purple Spanning Graph Problem*

Ahmad Biniaz† Prosenjit Bose* Ingo van Duijn ‡ Anil Maheshwari*
Michiel Smid*

August 30, 2016

Abstract

Consider a set of n points in the plane, each one of which is colored either red, blue, or purple. A red-blue-purple spanning graph (RBP spanning graph) is a graph whose vertices are the points and whose edges connect the points such that the subgraph induced by the red and purple points is connected, and the subgraph induced by the blue and purple points is connected. The minimum RBP spanning graph problem is to find an RBP spanning graph with minimum total edge length. First we consider this problem for the case when the points are located on a circle. We present an algorithm that solves this problem in $O(n^2)$ time, improving upon the previous algorithm by a factor of $O(n)$. Also, for the general case we present an algorithm that runs in $O(n^5)$ time, improving upon the previous algorithm by a factor of $O(n)$.

1 Introduction

Let S be a set of n points in the plane that is partitioned into $\{R, B, P\}$. The points of R are colored red, the points of B are colored blue, and the points of P are colored purple. A *red-blue-purple spanning graph* (RBP spanning graph) on S is a graph whose vertices are the points of S and whose edges connect the points such that each of the subgraphs induced by $R \cup P$ and by $B \cup P$ are connected. In other words, if we remove the red points then the resulting subgraph is connected, and if we remove the blue points then the resulting subgraph is connected. One may think of the purple points belonging to both the red set and the blue set. The *minimum RBP spanning graph problem* is to compute an RBP spanning graph that has minimum weight (total edge length). See [4, 5] for applications of this problem.

When the points of S are located on a line and given in sorted order, this problem can be solved in $O(n)$ time (see [4, 5]). If the points of S are located on a circle and given in circularly sorted order this problem can be solved in $O(n^3)$ time; specifically, it can be solved in $O(k^3+n)$ time, where k is the number of purple points (see [4, 5]). For points on a circle, in Sections 3 and 4, we show how to improve the running time to $O(k^2+n)$. In [4] it is claimed that the general case of this problem is NP-hard; this claim is based on a reduction from planar 3-SAT. They also presented an $O(n \log n)$-time $(1+\frac{\rho}{2})$-approximation algorithm for this problem, where ρ is the Steiner ratio. However, in [5] it is claimed that the NP-hardness reduction of [4] is incorrect,

*A preliminary version of this paper has appeared in CCCG 2016.

†School of Computer Science, Carleton University, Ottawa, Canada, ahmad.biniaz@gmail.com, {jit, anil, michiel}@scs.carleton.ca. Supported by NSERC.

‡MADALGO, department of Computer Science, Aarhus University, Denmark, ivd@cs.au.dk. MADALGO is supported in part by DNRF84.

and an $O(n^6)$-time exact algorithm for this problem is presented. This algorithm uses weighted matroid intersection. In Section 5 we show how to modify this algorithm to run in $O(n^5)$ time.

The input to the RBP spanning graph problem can be interpreted as a set of points in the plane and two primary color classes (red and blue in our case) such that each point belongs to one or more color classes (in our case the purple points belong to two classes). Recently, Akitaya *et al.* [2] considered this problem with more than two primary color classes; they showed that this version of the problem is NP-hard. For the case where the number of color classes is three, they presented a polynomial-time $(2 - \frac{1}{3+2\rho})$-approximation algorithm.

2 Properties of Minimum RBP Spanning Graphs

In this section we review some properties of minimum RBP spanning graphs. Given a graph G with vertex set S, and a set $S' \subseteq S$, we denote by $G[S']$ the subgraph of G that is induced by S'.

For three sets R, B, and P of red, blue, and purple points, respectively, we denote by $G^*(R, B, P)$ a minimum RBP spanning graph on $R \cup B \cup P$; this is denoted by G^* when the triple (R, B, P) is clear from the context. As in [4, 5] we classify the edges of G^* into red, blue, and purple. An edge is *red* if it connects two red points, or a red point and a purple point. An edge is *blue* if it connects two blue points, or a blue point and a purple point. An edge is *purple* if it connects two purple points. Note that G^* does not contain any edge between a red point and a blue point. The subgraph $G^*[P]$ that is induced by the purple points is acyclic, because otherwise we could remove a purple edge from a cycle and reduce the weight of G^* without destroying the connectivity of $G^*[R \cup P]$ and $G^*[B \cup P]$. The subgraph $G^*[R \cup P]$ (resp. $G^*[B \cup P]$) is a spanning tree because otherwise we could remove a red edge (resp. blue edge) from a cycle without affecting the connectivity of $G^*[B \cup P]$ (resp. $G^*[R \cup P]$). We refer to $G^*[R \cup P]$ as the *red tree* and to $G^*[B \cup P]$ as the *blue tree*.

Every red edge in G^* is also an edge of a minimum spanning tree of $R \cup P$, because otherwise we could replace it by another red or purple edge of smaller weight. The corresponding statement holds for the blue edges. Thus, the red edges of G^* do not cross each other, and the blue edges of G^* do not cross each other. The corresponding statements do not hold for purple edges. There can be purple edges in G^* that are not present in any minimum spanning tree of the purple points. Moreover, a purple edge in G^* can cross $\Theta(|P|)$ other purple edges [4, 5]. In [4, 5] it is shown that the maximum degree of a purple point in G^* is at most 18 and the maximum degree of a red point or a blue point is at most 6. Moreover, there exists an optimal graph in which the maximum degree of every purple point is at most 15 and the maximum degree of every red point or blue point is at most 5. The proofs for these degree constraints are inherited from the proofs of maximum degree constraints of minimum spanning trees of a point set in the plane.

3 The Algorithm for Points on a Circle

Let S be a set of n points on a circle C that are colored red, blue, or purple. Let R, B, and P denote the set of red, blue, and purple points of S, respectively. Let k denote the number of purple points, i.e., $k = |P|$. The problem is to compute a minimum RBP spanning graph for S. Although for points in the plane, and even for points in convex position, a purple edge can be crossed by other purple edges, for points on a circle, purple edges cannot be crossed by other purple edges. Based on this, Hurtado *et al.* [4, 5] presented a dynamic programming algorithm that solves this problem in $O(k^3 + n)$ time. We use a similar dynamic programming approach and improve the running time to $O(k^2 + n)$. First we review a lemma from [4, 5].

Lemma 1 (see [4, 5]). *Let S be a set of points on a circle, each one of which is colored either red, blue, or purple. Let G^* be a minimum RBP spanning graph for S. Then the following statements holds.*

1. *No purple edge of G^* can cross any other edge of G^*.*
2. *No red or blue edge of G^* can cross any segment between two purple points (which are not necessarily connected by an edge in G^*).*
3. *For any purple point p in S, let p' be the point on the circle diametrically opposite to p, and let SC be any of the two closed semicircles containing both p and p'. Then in G^*, p has at most one purple neighbor in SC, and thus at most two purple neighbors in total.*

3.1 The Dynamic Programming Algorithm

In this section we give an overview of the dynamic programming algorithm presented in [4, 5]. Assume that the points of S are circularly sorted. Let $p_1, \ldots, p_k$ be the purple points in clockwise order. For any $1 \leqslant i \leqslant k$, let S_i be the set of red and blue points between p_i and p_{i+1}. Assume that all indices are taken modulo k.

Let G^* be a minimum RBP spanning graph for S. By Lemma 1 no edge of G^* that is incident to a point in S_i can cross segment p_ip_{i+1} (p_ip_{i+1} is not necessarily an edge of G^*). Thus, a solution for each set S_i can be computed independently. Moreover, this is analogous to the case when the points are on a line, and thus, it can be solved in linear time for all sets S_i.

For any two purple points p_i and p_j, Lemma 1 guarantees that if p_ip_j is an edge in G^* then it cannot be crossed by any other edge of G^*. This introduces two independent subproblems, one to the left of the oriented segment p_ip_j, and one to the right. Each subproblem has four different types PC, RC, BC, and NC. In the PC-type, p_i and p_j are connected in both red and blue subgraphs. In the RC-type, p_i and p_j are connected in the red subgraph but disconnected in the blue subgraph; any solution for this type must connect p_i and p_j in its blue subgraph. In the BC-type, p_i and p_j are disconnected in the red subgraph but connected in the blue subgraph. In the NC-type, p_i and p_j are neither connected in the red subgraph nor in the blue subgraph. The algorithm maintains four tables, PC, RC, BC, and NC, each of size $O(k^2)$, that are indexed by pairs of purple points. Each entry $[i, j]$ of each table, stores the length of a minimum RBP spanning graph of the corresponding type for the point set $\{p_i, p_{i+1}, \ldots, p_j\}$. Based on this, the length of an optimal solution can be found as

$$\min_{2 \leqslant j \leqslant k} \{PC[1, j] + NC[j, 1], NC[1, j] + PC[j, 1], RC[1, j] + BC[j, 1], BC[1, j] + RC[j, 1]\}.$$

Let $\Lambda \in \{P, R, B, N\}$. The entries of each table are filled in order, so that when it is time to compute the value of an entry $\Lambda[i, j]$, all the entries corresponding to smaller problems, i.e., subproblems introduced by purple pairs to the left of the oriented segment p_ip_j, have already been computed. In order to fill entry $\Lambda C[i, j]$, where $1 \leqslant i < j \leqslant k$, the following two cases are considered, and the one with minimum cost will be stored in $\Lambda C[i, j]$. See [4, 5] for more details.

1. p_i is connected to some purple point(s) in an optimal solution of the subproblem (i, j); recall that by Lemma 1, p_i can be connected to at most two purple points. Let p_h be the one in the sequence $p_{i+1}, \ldots, p_j$ that is closer to p_j, see Figure 1(a). Note that p_i and p_h are connected in both red and blue subgraphs. Therefore, p_h and p_j must be connected in the same way as p_i and p_j. Since we do not know p_h, we try all possible candidates and keep the one minimizing the cost, i.e., $\Lambda C[i, j] = \min_{i+1 \leqslant h \leqslant j} \{PC[i, h] + \Lambda C[h, j] + |p_ip_h|\}$. This case takes $O(j - i)$ time.

Figure 1: Solving subproblem (i, j): (a) p_i is connected to a purple point p_h, and (b) p_i is not connected to any purple point.

2. p_i is not connected to other purple points in any optimal solution of the subproblem (i, j). Now Consider p_{i+1}. By Lemma 1, in an optimal solution no edge can cross the segment $p_i p_{i+1}$. Since no purple edge is incident to p_i, the segment $p_{i+1}p_j$ cannot be crossed either; see Figure 1(b). Therefore, an optimal solution for the subproblem (i, j) can be computed by combining the solutions associated to the subproblems $(i, i+1)$ and $(i+1, j)$. See [4, 5] for more details. This case takes $O(1)$ time.

Based on the description above, the total running time of the algorithm is $O(k^3 + n)$.

3.2 Improving the Running Time

In this section we show how to improve the running time of the algorithm presented in Section 3.1 to $O(k^2 + n)$. First we prove Lemma 2 which plays an important role in this regard.

A *chord* of a circle is a straight line segment whose endpoints lie on the circle. For any two points p and q on C let $SC(p, q)$ denote the smaller arc of C that is determined by the chord pq.

Lemma 2. *Let S be a set of red, blue, and purple points located on a circle. Let P be the set of purple points. Let a and b be any two points of P such that $SC(a, b)$ contains at least two points of $P \setminus \{a, b\}$. Let $a^*, b^* \in P$ be the purple neighbors of a and b on $SC(a, b)$, respectively.*

1. *If $|aa^*| + |bb^*| < |ab|$, then ab does not belong to any minimum RBP spanning graph for S.*

2. *If $|aa^*| + |bb^*| = |ab|$, then there exists a minimum RBP spanning graph of S that does not contain ab.*

Proof. First we prove statement 1 of the lemma. The proof is by contradiction. Assume there exists a minimum RBP spanning graph G^* for S that contains ab. Recall that G^* consists of a red tree and a blue tree; moreover, the purple edges of G^* belong to both trees. Let R_a and R_b be the two red trees obtained by removing ab from G^*, such that $a \in R_a$ and $b \in R_b$. Let B_a and B_b be the two blue trees obtained in a similar way.

Claim 1: *It is not possible to have $a^* \in R_b$ and $b^* \in R_a$, or $a^* \in B_b$ and $b^* \in B_a$.*

We prove this claim for the case where $a^* \in R_b$ and $b^* \in R_a$; the proof for the other case is similar. By Lemma 1 (item 2) no red edge or blue edge can "jump" over a^* or b^*. Thus, in order to have $a^* \in R_b$ and $b^* \in R_a$ there must be two purple edges in G^* that cross; this contradicts Lemma 1 (item 1). This proves the claim.

By Claim 1, $a^* \in R_a$ or $b^* \in R_b$. Without loss of generality assume that $a^* \in R_a$. If $a^* \in B_a$, then by replacing the edge ab in G^* with the purple edge a^*b we obtain a valid RBP

spanning graph that is smaller than G^* (note that a^*b is shorter than ab); see Figure 2(a). This contradicts the minimality of G^*. Assume that $a^* \in B_b$; see Figure 2(b). Then by Claim 1, we have $b^* \in B_b$. If $b^* \in R_b$, then by replacing the edge ab with the purple edge ab^* we obtain a valid RBP spanning graph that is smaller than G^*; see Figure 2(b). This contradicts the minimality of G^*. Assume that $b^* \in R_a$; see Figure 2(c). Then, by replacing ab with aa^* and bb^* we obtain a valid RBP spanning graph that is smaller than G^*. This contradicts the minimality of G^*.

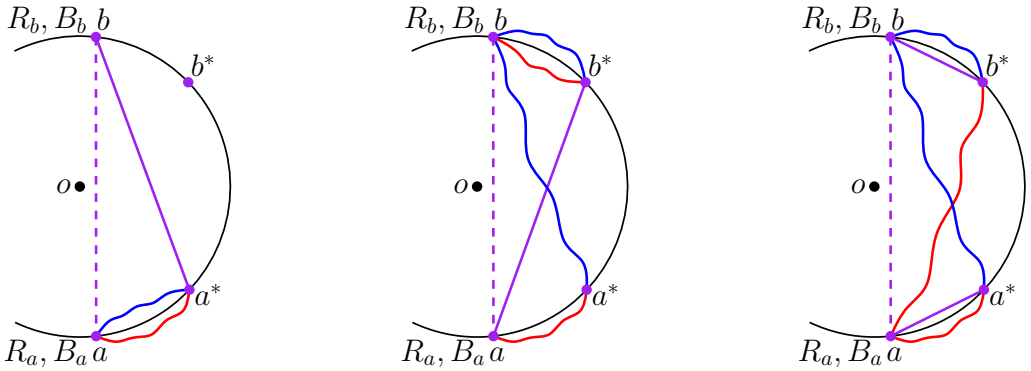

Figure 2: Proof of Lemma 2: (a) $a^* \in R_a$ and $a^* \in B_a$. (b) $a^* \in R_a$ and $a^* \in B_b$. (c) $a^* \in B_b$ and $b^* \in R_a$.

Now we prove statement 2 of the lemma. As we have seen in the proof of statement 1, if $|aa^*| + |bb^*| = |ab|$, in all cases we obtain an RBP spanning graph that is smaller than G^*, except for the case when we replace ab with aa^* and bb^*. Let G' be the graph that is obtained after replacing all such kind of edges. Since $|aa^*| + |bb^*| = |ab|$, G' has a weight equal to the weight of G^*. Moreover, G' does not contain ab. Thus, G' is a spanning graph that satisfies statement 2 of the lemma. □

We prove the following theorem in Section 4.

Theorem 1. *Let P be a set of points on a circle C. Let E_2 be the set of edges that contains an edge ab if and only if $a, b \in P$ and $|aa^*| + |bb^*| > |ab|$, where a^* and b^* are two points of P that are neighbors of a and b on the smaller arc of C that is determined by the chord ab, respectively. Then, no three edges of E_2 can pairwise cross.*

Theorem 2. *Let S be a set of n points located on a circle that are angularly sorted, and each one of which is colored either red, blue, or purple. Let k be the number of purple points. Then, a minimum red-blue-purple spanning graph on S can be computed in $O(k^2 + n)$ time.*

Proof. We define three sets of edges, E_0, E_1, and E_2, on the purple points as follows. Let a and b be any pair of purple points. If $SC(a, b)$ has no point of $P \setminus \{a, b\}$ then add ab to E_0. If $SC(a, b)$ contains exactly one point of $P \setminus \{a, b\}$ then add ab to E_1. Let E_2 be the set of purple edges that is defined in the statement of Theorem 1. Let $E_P = E_0 \cup E_1 \cup E_2$. As a consequence of Lemma 2 there exists an optimal solution in which all the purple edges belong to E_P. Thus, in case 1 of the dynamic programming algorithm, instead of looking at all pairs (p_i, p_h) it is enough to only consider the pairs (p_i, p_h) that are connected by an edge in E_P. Each pair (p_i, p_h) is considered only for the subproblems that have p_i or p_h as an endpoint; the number of such subproblems is $O(k)$. Thus, the total time we spend for case 1 is $O(k|E_P|)$. Therefore the total running time of the algorithm is $O(k|E_P| + n)$.

Note that E_P can be computed in $O(k^2)$ time in the preprocessing phase. We are going to show that $|E_P| = O(k)$; this will complete the proof of the theorem. Each of E_0 and E_1 contains $k = |P|$ edges. By Theorem 1 no three edges of E_2 pairwise cross. Agarwal *et al.* [1] have shown that any graph with n vertices that can be drawn in the plane such that no three edges pairwise cross, has $O(n)$ edges. Thus, E_2 has $O(k)$ edges. Therefore, $|E_P| = O(k)$. □

4 Proof of Theorem 1

In this section we prove Theorem 1. First we prove some lemmas that will be used in the proof of the theorem.

4.1 Preliminary Results

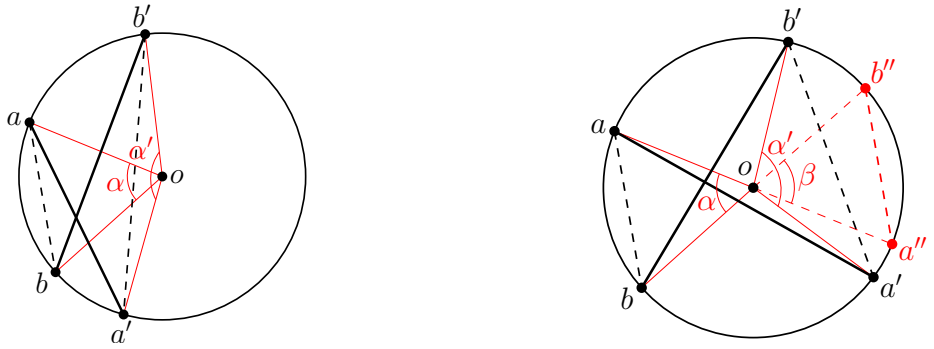

Figure 3: Proof of Lemma 3: (a) o does not lie in convex quadrilateral a, b, a', b', and (b) o lies in convex quadrilateral a, b, a', b'.

Lemma 3. *Let aa' and bb' be two intersecting chords of a circle C such that the center of C is to the left of the oriented segment aa' and to the right of the oriented segment bb'. Then, $|ab| < |a'b'|$.*

Proof. Let o be the center of C. Let $\alpha = \angle aob$ and $\alpha' = \angle a'ob'$. The triangles $\triangle abo$ and $\triangle a'b'o$ are isosceles. Based on this and assuming that the radius of C is 1, we have $|ab| = 2\sin\left(\frac{\alpha}{2}\right)$ and $|a'b'| = 2\sin\left(\frac{\alpha'}{2}\right)$. See Figure 3. Let Q be the convex quadrilateral with vertices a, b, a', and b'. If o does not lie in Q (see Figure 3(a)), then we have $\alpha < \alpha'$. This implies that $|ab| < |a'b'|$. Assume o lies in Q; see Figure 3(b). Let a'' and b'' be two points on C such that aa'' and bb'' are two diameters of C. Let $\beta = \angle a''ob''$. Note that $\beta = \alpha$. Moreover, we have $\beta < \alpha'$. This implies that that $\alpha < \alpha'$, and consequently $|ab| < |a'b'|$. □

Lemma 4. *Let b, a, and p be three points on a circle C, in clockwise order, such that the center of C is to the right side of the oriented segment bp. Let b'' be the point such that bb'' is a diameter of C. Let p' be a point on $SC(p, b'')$. Then $|ap'| - |ap| > |bp'| - |bp|$.*

Proof. Refer to Figure 4. Let $C(a, |ap|)$ be the circle of radius $|ap|$ that is centered at a, and let $C(b, |bp|)$ be the circle of radius $|bp|$ that is centered at b. Since a, p, and p' are on the same side of the line through bb'', we have $|ap'| > |ap|$ and $|bp'| > |bp|$. Let a' be the intersection point of ap' with $C(a, |ap|)$, and let b' be the intersection point of bp' with $C(b, |bp|)$. Then $|aa'| = |ap|$ and $|bb'| = |bp|$. Thus, $|ap'| - |ap| = |a'p'|$ and $|bp'| - |bp| = |b'p'|$. In order to

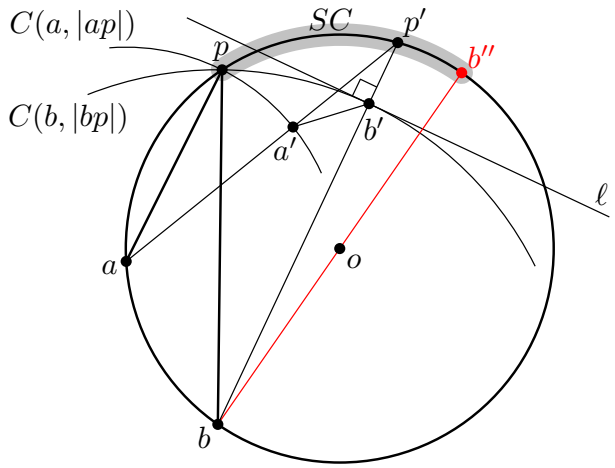

Figure 4: Proof of Lemma 4.

prove the statement of the lemma it suffices to show that $|a'p'| > |b'p'|$. Let ℓ be the line that is tangent to $C(b, |bp|)$ at b'. Observe that a' and p' are on different sides of ℓ. Thus, in triangle $\triangle a'b'p'$, the angle $\angle a'b'p'$ is at least $\frac{\pi}{2}$. This implies that $a'p'$ is the longest side of $\triangle a'b'p'$. Therefore, $|a'p'| > |b'p'|$; this completes the proof of the lemma. □

Lemma 4 can be restated as follows. If we fix the position of a and b, then by moving p towards b'', the length of ap increases more than the length of bp.

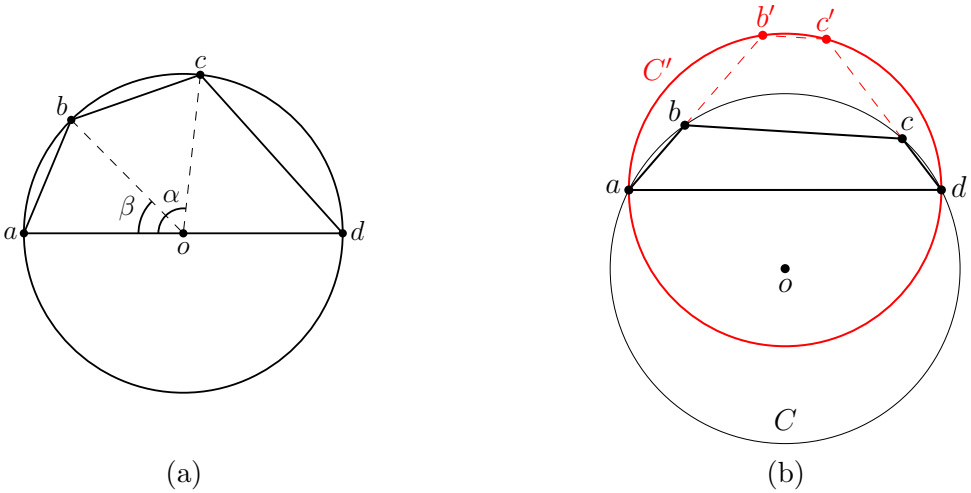

Figure 5: (a) Proof of Lemma 5 and Lemma 6. (b) Proof of Lemma 6.

Lemma 5. *Let $0 < \alpha \leqslant \pi$ be fixed, and let C be a circle that is centered at o. Let a, b, and c be three points on C, in clockwise order, such that $\angle aoc = \alpha$. Then, $|ab| + |bc|$ is maximum when $\angle aob = \angle boc = \frac{\alpha}{2}$.*

Proof. Let $f = |ab| + |bc|$, and let $\beta = \angle aob$. See Figure 5(a). Since the triangles $\triangle aob$ and $\triangle aoc$ are isosceles, we have $f = 2\sin\left(\frac{\beta}{2}\right) + 2\sin\left(\frac{\alpha-\beta}{2}\right)$. By taking the derivative of f with respect to β, we can see that f is maximum when $\beta = \frac{\alpha}{2}$, and thus, $\angle aob = \angle boc = \frac{\alpha}{2}$. □

The following is a corollary of Lemma 5.

Corollary 1. *Let C be a circle that is centered at o. Let a, b, and c be three points on C, in clockwise order, such that $\angle aoc \leqslant \frac{2\pi}{3}$. Then $|ab| + |bc| \leqslant |ao| + |co|$.*

Proof. By Lemma 5, $|ab| + |bc|$ is maximum when $\angle aob = \angle boc$, and thus, both these angles are at most $\frac{\pi}{3}$. This implies that $|ab| \leqslant |ao|$ and $|bc| \leqslant |co|$, which proves the claim. □

The following theorem is a restatement of Theorem 7.11 in [3].

Theorem 3 (See [3])**.** *If C_1 and C_2 are convex polygonal regions with $C_1 \subseteq C_2$, then the length of the boundary of C_1 is at most the length of the boundary of C_2.*

Lemma 6. *Let a, b, c, and d be four points on a circle C, in clockwise order, such that the center of C is on or to the right side of the oriented segment ad. Then $|ab| + |bc| + |cd| \leqslant \frac{3}{2} \cdot |ad|$.*

Proof. Without loss of generality assume C is centered at o and has radius 1. We consider two cases: (i) o is on ad, and (ii) o is not on ad. First, we prove case (i). Then, we show how to reduce case (ii) to case (i). Assume o is on ad, that is, ad is a diameter of C, and thus, $|ad| = 2$. See Figure 5(a). If we fix the position of c on C, then by Lemma 5, $|ab| + |bc|$ is maximum when $\angle aob = \angle boc$. If we fix the position of b on C, then by Lemma 5, $|bc| + |cd|$ is maximum when $\angle boc = \angle cod$. Therefore, $|ab| + |bc| + |cd|$ is maximum when $\angle aob = \angle boc = \angle cod = \frac{\pi}{3}$, and thus, $|ab| = |bc| = |cd| = |ao| = |od|$. This implies the statement of the lemma for case (i).

Now we show how to handle case (ii). Assume o is not on ad, and thus, ad is not a diameter of C. We show how to reduce this case to case (i). Follow Figure 5(b). Let C' be the circle with diameter ad. Since C and C' intersect only at the two points a and d, we argue that b and c are in the interior of C'. Extend ab and dc to intersect C' at b' and c', respectively. Now we consider two cases depending on whether bb' and cc' intersect or not. In the former case, let o' be the intersection point of bb' and cc'. By Theorem 3 we have $|ab| + |bc| + |cd| \leqslant |ao'| + |o'd|$. Since o' is in the interior of C' then $|ao'| + |o'd| \leqslant \sqrt{2} \cdot |ad|$; and we are done with this case. In the latter case, by Theorem 3 we have $|ab| + |bc| + |cd| \leqslant |ab'| + |b'c'| + |c'd|$. As we have seen in case (i), $|ab'| + |b'c'| + |c'd| \leqslant \frac{3}{2} \cdot |ad|$; which completes the proof of the lemma. □

4.2 Proof of Theorem

Recall that P is a set of points on a circle C. The edge set E_2 contains an edge ab if and only if $a, b \in P$ and $|aa^*| + |bb^*| > |ab|$, where a^* and b^* are the two points of P that are neighbors of a and b on the smaller arc of C that is determined by the chord ab, respectively. Without loss of generality assume C is centered at o and has radius 1. Based on the definition of E_2, the following observation is valid.

Observation 1. *For any edge $ab \in E_2$, we have $|aa^*| > \frac{1}{2} \cdot |ab|$ or $|bb^*| > \frac{1}{2} \cdot |ab|$.*

Corollary 2. *For any edge $ab \in E_2$, we have $|a^*b^*| < \frac{1}{2} \cdot |ab|$.*

Proof. Since $ab \in E_2$, we have $|aa^*| + |bb^*| > |ab|$. By Lemma 6 we have $|aa^*| + |a^*b^*| + |b^*b| \leqslant \frac{3}{2} \cdot |ab|$ where a, a^*, b^*, and b play the role of a, b, c, and d, respectively. This implies that $|a^*b^*| < \frac{1}{2} \cdot |ab|$. □

Now we have all the tools that we need to prove Theorem 1. For the sake of contradiction assume that three edges aa', bb', and cc' of E_2 are pairwise crossing. Observe that if we remove all points of P except a, b, c, a', b', c', and then recompute E_2, the edges aa', bb', and cc' will remain in E_2. Thus, without loss of generality we assume that $P = \{a, b, c, a', b', c'\}$. Moreover, assume a, b, c, a', b', c' appear in clockwise order on C. Let $\triangle$ be the triangle whose vertices are the intersection points of aa', bb', and cc'. We differentiate between the following two cases: (i)

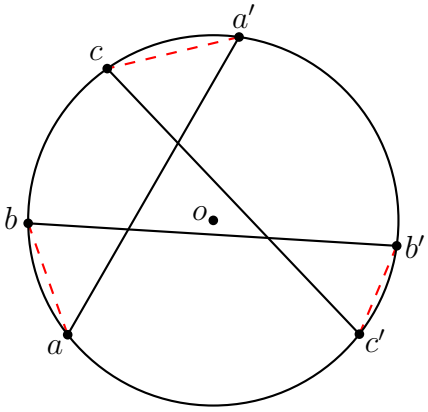

Figure 6: Proof of case (i) in Theorem 1.

o is in the interior of $\triangle$, and (ii) o is not in the interior of $\triangle$. We will get contradictions in both cases.

First we handle case (i). Refer to Figure 6. Since aa', bb', and cc' are edges of E_2, we have $|ab| + |a'c| > |aa'|$, $|ab| + |b'c'| > |bb'|$, and $|b'c'| + |a'c| > |cc'|$. By adding up these three inequalities, we get

$$|ab| + |a'c| + |b'c'| > \frac{|aa'| + |bb'| + |cc'|}{2}. \tag{1}$$

By Lemma 3 we have $|ab| < |a'b'|$, $|a'c| < |ac'|$, and $|b'c'| < |bc|$. Adding up these three inequalities implies

$$|ab| + |a'c| + |b'c'| < |a'b'| + |ac'| + |bc|. \tag{2}$$

In view of Corollary 2, we have $|bc| < \frac{1}{2} \cdot |aa'|$, $|ac'| < \frac{1}{2} \cdot |bb'|$, and $|a'b'| < \frac{1}{2} \cdot |cc'|$. This implies

$$|bc| + |ac'| + |a'b'| < \frac{|aa'| + |bb'| + |cc'|}{2}.$$

This and Inequality (2) imply

$$|ab| + |a'c| + |b'c'| < \frac{|aa'| + |bb'| + |cc'|}{2},$$

which contradicts Inequality (1); this is a contradiction for case (i).

Now we are going to handle case (ii) where o is not in the interior of the triangle formed by the intersection of aa', bb', and cc'. Without loss of generality assume that o is on or to the right side of all the oriented segments aa', bb', and cc'; see Figure 7(a). Since aa', bb', and cc' are edges of E_2, we have $|ab| + |a'c| > |aa'|$, $|bc| + |a'b'| > |bb'|$, and $|a'c| + |b'c'| > |cc'|$. By adding up these three inequalities, we get

$$|ab| + |bc| + |a'b'| + |b'c'| + 2|a'c| > |aa'| + |bb'| + |cc'|. \tag{3}$$

Let a'' and c'' be the two points on C such that $a'a''$ and cc'' are diameters of C. By Lemma 4, $|a''b| - |ab| > |a''a'| - |aa'|$ and $|b'c''| - |b'c'| > |cc''| - |cc'|$. By adding these two inequalities to Inequality (3) we get

$$|a''b| + |bc| + |a'b'| + |b'c''| + 2|a'c| > |a''a'| + |bb'| + |cc''|. \tag{4}$$

Thus, if o is on or to the right side of all the oriented segments aa', bb', and cc', then Inequality (4) is valid. In fact, Inequality (4) is the same as Inequality (3) where a'' and c'' play the role of a

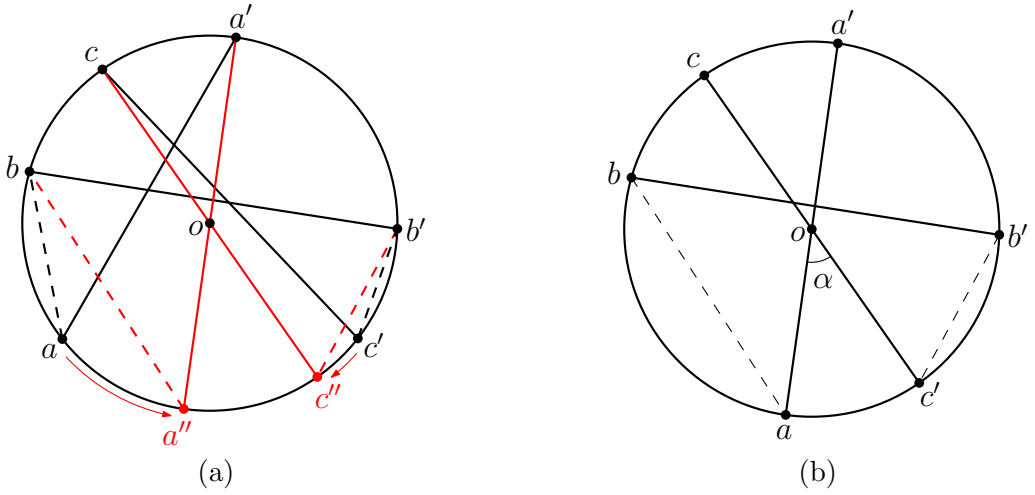

Figure 7: Proof of case (ii) in Theorem 1.

and c', respectively. Therefore, without loss of generality, from now on we assume that a is on a'' and c' is on c'', that is, aa' and cc' are two diameters of C.

Let $\alpha = \angle aoc' = \angle coa'$. We claim that $\alpha \leqslant \frac{\pi}{3}$. Assume $\alpha > \frac{\pi}{3}$. This implies that $\angle aoc \leqslant \frac{2\pi}{3}$ and $\angle a'oc' \leqslant \frac{2\pi}{3}$. By Corollary 1 we have $|ab| + |bc| \leqslant |ao| + |oc|$ and $|a'b'| + |b'c'| \leqslant |a'o| + |oc'|$. As a consequence of Corollary 2, for edge bb' we have $2|a'c| < |bb'|$. By adding these three inequalities we get

$$|ab| + |bc| + |a'b'| + |b'c'| + 2|a'c| < |ao| + |oc| + |a'o| + |oc'| + |bb'| = |aa'| + |bb'| + |cc'|,$$

which contradicts Inequality (3). Therefore, $\alpha \leqslant \frac{\pi}{3}$.

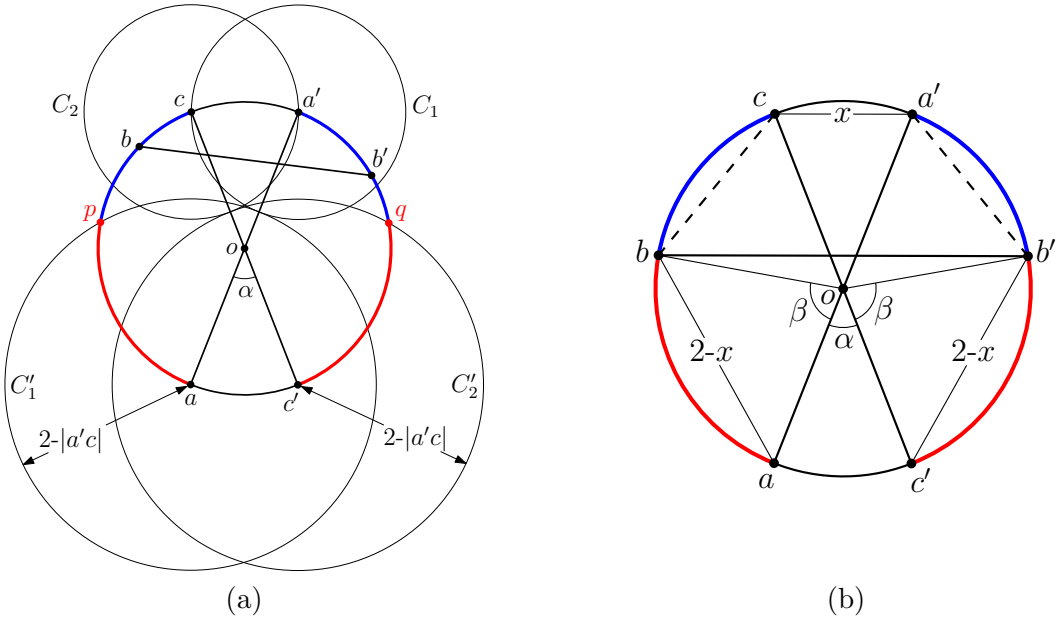

Figure 8: Proof of case (ii) in Theorem 1.

Recall that C has radius 1, and thus, $|aa'| = |cc'| = 2$. Consider two circles $C_1 = C(a', |a'c|)$ and $C_2 = C(c, |a'c|)$. Let C'_1 be the circle that is centered at a and touches C_1, i.e., C'_1 has radius $2 - |a'c|$. Similarly, let C'_2 be the circle that is centered at c' and touches C_2. See Figure 8(a).

Let $\widehat{ac}$ (resp. $\widehat{a'c'}$) be the smaller arc of C that has endpoints a to c (resp. a' and c'). Let p be the intersection point of C_1' and $\widehat{ac}$. Let $\widehat{ap}$ and $\widehat{pc}$ be the two sub-arcs of $\widehat{ac}$. Similarly, let q be the intersection point of C_2' and $\widehat{a'c'}$, and let $\widehat{a'q}$ and $\widehat{qc'}$ be the two sub-arcs of $\widehat{a'c'}$.

If b is in the interior of $\widehat{ap}$ then $|ab| + |a'c| < 2 = |aa'|$, which contradicts the existence of aa' in E_2. Thus $b \in \widehat{pc}$, and similarly, $b' \in \widehat{a'q}$. We are going to show that $|bc| + |a'b'| \leqslant |bb'|$; this will contradict the existence of bb' in E_2.

Since $bb' \in E_2$ we have $|bc| + |a'b'| > |bb'|$. By Lemma 4 if we move b towards p, then $|bc|$ increases more than $|bb'|$. Similarly, if we move b' towards q, then $|a'b'|$ increases more than $|bb'|$. Therefore, $|bc| + |a'b'| > |bb'|$ holds after moving b to p, and b' to q. Thus, from now we assume $b = p$ and $b' = q$. See Figure 8(b). Note that all the triangles $\triangle aob$, $\triangle boc$, $\triangle coa'$, $\triangle a'ob'$, $\triangle b'oc'$, and $\triangle bob'$ are isosceles. Let $x = |a'c|$, and thus, $|ab| = |b'c'| = 2 - |a'c| = 2 - x$. Note that $x = 2\sin\left(\frac{\alpha}{2}\right)$. Let $\beta = \angle aob = \angle b'oc'$. Then, $\beta = 2\arcsin\left(\frac{2-x}{2}\right) = 2\arcsin\left(1 - \sin\left(\frac{\alpha}{2}\right)\right)$. Note that $\angle bob' = 2\pi - \alpha - 2\beta$. Thus,

$$|bb'| = 2\sin\left(\pi - \frac{\alpha}{2} - \beta\right) = 2\sin\left(\frac{\alpha}{2} + \beta\right).$$

Moreover, $\angle boc = \angle a'ob' = \pi - \alpha - \beta$. Thus,

$$|bc| = |a'b'| = 2\sin\left(\frac{\pi - \alpha - \beta}{2}\right) = 2\cos\left(\frac{\alpha + \beta}{2}\right).$$

Now we show that $|bc| + |a'b'| \leqslant |bb'|$, which contradicts the existence of bb' in E_2. In order to show this, it suffices to prove that

$$4\cos\left(\frac{\alpha + \beta}{2}\right) \leqslant 2\sin\left(\frac{\alpha}{2} + \beta\right), \tag{5}$$

where $\beta = 2\arcsin\left(1 - \sin\left(\frac{\alpha}{2}\right)\right)$, for all $0 < \alpha \leqslant \frac{\pi}{3}$. Inequality (5) simplifies to

$$2\sqrt{1 - \sin^2\left(\frac{\alpha}{2}\right)}\sqrt{1 - \left(1 - \sin\left(\frac{\alpha}{2}\right)\right)^2} + 2\left(1 - \sin\left(\frac{\alpha}{2}\right)\right)^2 + \sin\left(\frac{\alpha}{2}\right) \leqslant 3, \tag{6}$$

where $0 < \alpha \leqslant \frac{\pi}{3}$. Let $u = \sin\left(\frac{\alpha}{2}\right)$. Then, Inequality (6) simplifies to

$$4u^2 - 4u + 1 \geqslant 0, \tag{7}$$

where $0 \leqslant u \leqslant \frac{1}{2}$; it is easy to verify that Inequality (7) is valid in this range of u. This contradicts the fact that $|bc| + |a'b'| > |bb'|$, and hence the existence of bb' in E_2; this is a contradiction for case (ii). This completes the proof of Theorem 1.

5 The Algorithm for the General Case

In this section we consider the general case of the problem. The input to the problem consists of three pairwise disjoint sets R, B, and P of points in the plane that are colored red, blue, and purple, respectively. The problem is to compute a minimum RBP spanning graph for $R \cup B \cup P$. In [4] it was claimed that this problem is NP-hard. However, the NP-hardness reduction turned out to be incorrect, and a polynomial time algorithm for this problem is presented in [5]. The algorithm is greedy and based on matroid theory. In fact it is based on the existence of an efficient algorithm for the weighted matroid intersection problem. The algorithm runs in $O(n^6)$ time where n is the total number of points. In Section 5.1 we give an overview of the algorithm presented in [5]. We also add more details on the weighted matroid intersection algorithm. In Section 5.2 we show how to improve the running time to $O(k^5 + k^3 n + kn^2)$, where k is the number of purple points.

5.1 Overview of the Algorithm

Let $S = R \cup B \cup P$. Let E be the set of edges of the complete geometric graph on S except the edges that connect a red point to a blue point. Let $m = |E|$ and $n = |S|$. Note that $m = \Theta(n^2)$. Let $G = (S, E)$ be the edge-weighted graph with vertex set S and edge set E, where the weight $w(e)$ of an edge $e \in E$ is its Euclidean length. Let $G^* = (S, X^*)$ be any minimum RBP spanning graph on S. Note that X^* is a subset of E.

Let $\mathcal{I}_r$ be the collection of all subsets of E that form a forest on $R \cup P$. Similarly, let $\mathcal{I}_b$ be the collection of all subsets of E that form a forest on $B \cup P$. The pairs $(E, \mathcal{I}_r)$ and $(E, \mathcal{I}_b)$ are matroids (and known as *graphic matroids*). See [6, Chapter 39] for the basic concepts of matriod theory. The elements of $\mathcal{I}_r$ are called *independent sets*. The independents sets of $\mathcal{I}_r$ that have the maximum number of elements are called *bases*. Thus, the bases of $\mathcal{I}_r$ are the spanning trees for $R \cup P$. Let $\mathcal{B}_r$ be the set of bases of $\mathcal{I}_r$. Similarly let $\mathcal{B}_b$ be the set of bases of $\mathcal{I}_b$. We define supersets of $\mathcal{B}_r$ and $\mathcal{B}_b$ as follows:

$$\mathcal{Q}_r = \{X \subseteq E : X \text{ contains some } B \in \mathcal{B}_r\},$$
$$\mathcal{Q}_b = \{X \subseteq E : X \text{ contains some } B \in \mathcal{B}_b\}.$$

Note that $\mathcal{Q}_r$ (resp. $\mathcal{Q}_b$) is the set of all spanning graphs of $R \cup P$ (resp. $B \cup P$). Then, the minimum RBP spanning graph problem is formulated as follows:

$$\text{(P0)} \quad \text{Minimize} \sum_{e \in X} w(e) \quad \text{subject to } X \in \mathcal{Q}_r \cap \mathcal{Q}_b.$$

Let $\overline{\mathcal{I}}_r = \{E \setminus Y : Y \in \mathcal{Q}_r\}$ and $\overline{\mathcal{I}}_b = \{E \setminus Y : Y \in \mathcal{Q}_b\}$. Then, $(E, \overline{\mathcal{I}}_r)$ and $(E, \overline{\mathcal{I}}_b)$ are matroids that are dual of $(E, \mathcal{I}_r)$ and $(E, \mathcal{I}_b)$, respectively. Consider the following maximization problem

$$\text{(P1)} \quad \text{Maximize} \sum_{e \in Y} w(e) \quad \text{subject to } Y \in \overline{\mathcal{I}}_r \cap \overline{\mathcal{I}}_b.$$

Note that the complement of any element Y in $\overline{\mathcal{I}}_r \cap \overline{\mathcal{I}}_b$ is a valid RBP spanning graph that belongs to $\mathcal{Q}_r \cap \mathcal{Q}_b$ and spans both $R \cup P$ and $B \cup P$. On the other hand, any element X in $\mathcal{Q}_r \cap \mathcal{Q}_b$ is a valid RBP spanning graph whose complement belongs to $\overline{\mathcal{I}}_r \cap \overline{\mathcal{I}}_b$. Thus, the complement of a solution for (P0) is a solution for (P1) and vice versa, that is, both problems are equivalent. The problem (P1) is an instance of the weighted matroid intersection problem in which we are looking for a common element of $\overline{\mathcal{I}}_r$ and $\overline{\mathcal{I}}_b$ that has maximum-weight; for this problem a polynomial-time algorithm exists. In the following paragraph we give a brief description of an algorithm that finds a maximum-weight independent set Y^* in $\overline{\mathcal{I}}_r \cap \overline{\mathcal{I}}_b$. Then, $X^* = E \setminus Y^*$ will be a solution for (P0).

The general idea of the (maximum) weight matroid intersection algorithm is as follows (see [6, Chapter 41] for more details). Let I be a common independent set that has the maximum weight among all the common independents sets of size $|I|$. The algorithm computes a common independent set I', with $|I'| = |I| + 1$, that has the maximum weight among all the common independent sets with $|I| + 1$ elements. Let M be the maximum-size of a common independent set of the two matroids. Since $I_0 = \emptyset$ is a maximum-weight common independent set of size zero, the algorithm iteratively finds common independent sets $I_0, I_1, \ldots, I_M$, such that I_i is a maximum-weight common independent set of size i, i.e. $|I_i| = i$. Note that I_M is a maximum-weight common independent set of maximum-size. Taking one among $I_0, \ldots, I_M$ of maximum weight, we have a maximum-weight common independent set.

Since the minimum-size of an element in $\mathcal{Q}_r \cap \mathcal{Q}_b$ is $n-1$, the maximum-size M of a common independent set of $\overline{\mathcal{I}}_r$ and $\overline{\mathcal{I}}_b$ is $|E|-(n-1) = m-n+1$. Therefore by computing $I_0, \ldots, I_{m-n+1}$

one can obtain a maximum-weight element of $\overline{\mathcal{I}}_r \cap \overline{\mathcal{I}}_b$; the algorithm presented in [5] runs this way. This algorithm solves $m-n+1 = O(n^2)$ instances of weighted matriod intersection, where each instance can be solved in $O(m^2 + m \log m) = O(n^4)$ time. Thus, the total running time of their algorithm is $O(n^6)$. See [5] for more details on the time complexity analysis and for an interpretation of this algorithm in terms of the original problem (P0).

5.2 Improving the Running Time

In this section we show how to improve the running time of the algorithm described in Section 5.1 to $O(k^5 + k^3 n + kn^2)$. This improvement is obtained by modifying the algorithm in two ways: (i) by decreasing the total number of edges that have to be considered, and (ii) by reducing the number of instances of the weighted matroid intersection problem. The *Gabriel graph* on a given set of points in the plane is defined to have an edge between any two input points p and q if the closed disk with diameter pq does not contain any other input point.

Lemma 7. *Every red edge (resp. blue edge) of any minimum RBP spanning graph on (R, B, P) belongs to the Gabriel graph with vertex set $R \cup P$ (resp. $B \cup P$).*

Proof. We prove the statement for the red edges; the proof for the blue edges is similar. Let G^* be a minimum RBP spanning graph. Take a red edge ab from G^*. Note that ab belongs to a Euclidean minimum spanning tree T with vertex set $R \cup B$. Let $D[a, b]$ be the closed disk that has ab as a diameter. Since ab belongs to T, $D[a, b]$ does not contain any point of $R \cup B$, because otherwise we could replace ab by another red or purple edge of smaller size which contradicts the minimality of T. This implies that ab is an edge of the Gabriel graph on $R \cup P$. □

Let E_R (resp. E_B) be the set of edges of the Gabriel graph with vertex set $R \cup P$ (resp. $B \cup P$). Let E_P be the set of purple edges between any pair of purple points. Note that $|E_R| = O(|R| + |P|)$, $|E_B| = O(|B| + |P|)$, and $|E_P| = O(|P|^2)$. Let $E' = E_R \cup E_B \cup E_P$, and $m' = |E'|$. Note that $m' = O(k^2 + n)$. Let $G^* = (S, X^*)$ be any minimum RBP spanning graph for S. As a consequence of Lemma 7, X^* is a subset of E'. Thus, in the algorithm of Section 5.1 it suffices to look at the matriods defined on E'. This improves the running time to $O(m' \cdot ((m')^2 + m' \log m')) = O(k^6 + k^4 n + k^2 n^2 + n^3)$. In the rest of this section we describe how to improve the running time further.

Lemma 8. *Let R, B, and P be pairwise disjoint sets of points in the plane that are colored red, blue, and purple, respectively. Then every minimum RBP spanning graph for $R \cup B \cup P$ has at most $|R| + |B| + 2|P| - 2$ edges.*

Proof. Let G^* be a minimum RBP spanning graph for $R \cup B \cup P$. The induced subgraphs $G^*[R \cup P]$ and $G^*[B \cup P]$ are trees and have $|R| + |P| - 1$ and $|B| + |P| - 1$ edges, respectively. Since the edge set of G^* is the union of the edge set of $G^*[R \cup P]$ and the edge set of $G^*[B \cup P]$, we conclude that G^* has at most $|R| + |B| + 2|P| - 2$ edges. □

Lemma 9. *Let S be a set of n points in the plane that are colored either red, blue, or purple. Then,*

1. *every RBP spanning graph for S has at least $n - 1$ edges.*
2. *there exists an RBP spanning graph for S that has $n - 1$ edges.*
3. *in any RBP spanning graph of S with $n - 1$ edges, the subgraph induced by the purple points is a tree.*

Proof. Since any RBP spanning graph is connected, it has at least $n-1$ edges. This proves the first statement.

To prove the second statement we construct an RBP spanning graph with $n-1$ edges. First we compute a spanning tree on the purple points, then we connect every other point (red or blue) to a purple point. The resulting graph is a valid RBP spanning graph that is a tree and and has $n-1$ edges.

In order to prove the third statement, let G be an RBP spanning graph of S, having $n-1$ edges. Since G is connected and has $n-1$ edges, it is a tree. Let P be the set of purple points of S. For the sake of contradiction assume there are two points p and q in P such that there is no path between them in $G[P]$, i.e., there is no purple path between p and q. Since G is an RBP spanning graph, there is a path between p and q in the red tree, and there is another path between p and q in the blue tree. This creates a cycle in G, which contradicts the fact that G is a tree. Thus, $G[P]$ is connected; moreover it is a tree. □

Let X_i^* denote the set of edges of a minimum RBP spanning graph with exactly i edges. As a consequence of Lemma 9 there exists a minimum RBP spanning graph with $|R|+|B|+|P|-1 = n-1$ edges; note that this is the smallest possible number of edges for any RBP spanning graph. By Lemma 8 any minimum RBP spanning graph has at most $|R|+|B|+2|P|-2 = n+k-2$ edges. Thus, the weight of X^* is equal to the smallest weight among the weights of X_i^* for all $n-1 \leqslant i \leqslant n+k-2$. Since the addition of edges maintains the RBP spanning property, we conclude that for every $i \in \{n-1, \ldots, n+k-2\}$ there exists an RBP spanning graph with exactly i edges. Based on Lemma 9, X_{n-1}^* can simply be computed by first computing the minimum spanning tree of the purple points, and then adding the red points, and then the blue points, in an optimal manner in a similar way as Prim's algorithm for minimum spanning trees.

One can easily modify the weighted matroid intersection algorithm as follows: having a maximum-weight common independent set I with $|I|$ elements, we can compute a maximum-weight common independent set I' with $|I'| = |I|-1$ elements. Let M' be the maximum-size of a common independent set of the two matroids. Therefore, if we have $I_{M'}$, the algorithm can be modified to first compute $I_{M'}, I_{M'-1}, \ldots, I_0$ and then take the one with maximum weight.

Since the minimum-size of an element in $\mathcal{Q}_r \cap \mathcal{Q}_b$ is $n-1$, the maximum-size M' of a common independent set of $\overline{\mathcal{I}}_r$ and $\overline{\mathcal{I}}_b$ is $|E'|-(n-1) = m'-n+1$. As described above, we can compute a minimum-weight element X_{n-1}^* in $\mathcal{Q}_r \cap \mathcal{Q}_b$ that has $n-1$ edges. The complement of X_{n-1}^* is a maximum-weight common independent set $I_{m'-n+1}$ of $\overline{\mathcal{I}}_r$ and $\overline{\mathcal{I}}_b$. Therefore by computing $I_{m'-n+1}, \ldots, I_{m'-n-k+2}$ and taking the one with maximum weight, we obtain a maximum-weight element of $\overline{\mathcal{I}}_r \cap \overline{\mathcal{I}}_b$. Note that in (P0) we compute $X_{n-1}^*, \ldots, X_{n+k-2}^*$ and take the one with the smallest weight as X^*. We solve $O(k)$ instances of the weighted matriod intersection problem, each of which can be solved in $O((m')^2 + m' \log m') = O(k^4 + k^2 n + n^2)$ time (recall that $m' = O(k^2+n)$). Thus, the total running time of the algorithm is $O(k^5 + k^3 n + k n^2)$. Since k is at most n, the running time of the algorithm is $O(n^5)$.

Recently, Akitaya *et al.* [2] showed that it is possible to solve this problem by computing a subset of red edges and a subset of blue edges of a minimum RBP spanning graph in advance in $O(n \log n)$ time, and then run the matroid intersection algorithm on a set m' of size $O(k^2)$. This, in turn, improves the running time of the algorithm to $O(k^5 + n \log n)$, however, the worst case running time of the algorithm is still $O(n^5)$. We summarize the result of the discussion of this section in the following theorem.

Theorem 4. *Let S be a set of n points in the plane, each one of which is colored either red, blue, or purple. Let k be the number of purple points. Then, a minimum red-blue-purple spanning graph on S can be computed in $O(k^5 + n \log n)$ time.*

References

[1] P. K. Agarwal, B. Aronov, J. Pach, R. Pollack, and M. Sharir. Quasi-planar graphs have a linear number of edges. *Combinatorica*, 17(1):1–9, 1997.

[2] H. A. Akitaya, M. Löffler, and C. D. Tóth. Multi-colored spanning graphs. *CoRR*, arXiv:1608.07056 [cs.CG], 2016. Also to appear in *24th International Symposium on Graph Drawing and Network Visualization* (GD), 2016.

[3] R. V. Benson. *Euclidean geometry and convexity.* McGraw-Hill, 1966.

[4] F. Hurtado, M. Korman, M. J. van Kreveld, M. Löffler, V. S. Adinolfi, R. I. Silveira, and B. Speckmann. Colored spanning graphs for set visualization. In *21st Int. Symp. on Graph Drawing*, pages 280–291, 2013.

[5] F. Hurtado, M. Korman, M. J. van Kreveld, M. Löffler, V. Sacristán, A. Shioura, R. I. Silveira, B. Speckmann, and T. Tokuyama. Colored spanning graphs for set visualization. *CoRR*, arXiv:1603.00580 [cs.CG], 2016. Also to appear in *Comput. Geom., special issue in Memoriam: Ferran Hurtado.*

[6] A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency.* Springer, 2003.