Colored Spanning Graphs for Set Visualization[⋆]

Ferran Hurtado[1], Matias Korman[1], Marc van Kreveld[2], Maarten Löffler[2], Vera Sacristán[1], Rodrigo I. Silveira[4,1], and Bettina Speckmann[3]

[1] Dept. de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Spain
{ferran.hurtado,matias.korman,vera.sacristan}@upc.edu

[2] Dept. of Computing and Information Sciences, Utrecht University, the Netherlands
{m.loffler,m.j.vankreveld}@uu.nl

[3] Dept. of Mathematics and Computer Science,
Technical University Eindhoven, The Netherlands
speckman@win.tue.nl

[4] Dept. de Matemática, Universidade de Aveiro, Portugal
rodrigo.silveira@ua.pt

Abstract. We study an algorithmic problem that is motivated by ink minimization for sparse set visualizations. Our input is a set of points in the plane which are either blue, red, or purple. Blue points belong exclusively to the blue set, red points belong exclusively to the red set, and purple points belong to both sets. A *red-blue-purple spanning graph* (RBP spanning graph) is a set of edges connecting 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.

We study the geometric properties of minimum RBP spanning graphs and the algorithmic problems associated with computing them. Specifically, we show that the general problem is NP-hard. Hence we give an $(\frac{1}{2}\rho+1)$-approximation, where ρ is the Steiner ratio. We also present efficient exact solutions if the points are located on a line or a circle. Finally we consider extensions to more than two sets.

1 Introduction

Visualizing sets and their elements is a recurring theme in information visualization. Sets arise in many application areas, as varied as social network analysis (grouping individuals into communities), linguistics (related words), or geography (related places). Among the oldest representations for sets are Venn diagrams [11] and their generalization, Euler diagrams. These representations are natural and effective for a small number of elements and sets. However, for larger numbers of sets and more complicated intersection patterns more intricate solutions are necessary. The last years have seen a steady

⋆ M.L. was supported by the Netherlands Organisation for Scientific Research (NWO) under grant 639.021.123. F. H., M. K., V. S. and R.I. S. were partially supported by ESF EUROCORES programme EuroGIGA, CRP ComPoSe: grant EUI-EURC-2011-4306, and by project MINECO MTM2012-30951. F. H., V. S. and R.I. S. were supported by project Gen. Cat. DGR 2009SGR1040. M. K. was supported by the Secretary for Universities and Research of the Ministry of Economy and Knowledge of the Government of Catalonia and the European Union. R. I. S. was funded by the FP7 Marie Curie Actions Individual Fellowship PIEF-GA-2009-251235 and by FCT through grant SFRH/BPD/88455/2012.

stream of developments in this direction, both for the situation where the location of set elements can be freely chosen and for the important special case that elements have to be drawn at particular fixed positions (for example, restaurant locations on a city map).

Our paper is motivated by some recent approaches that use very sparse enclosing shapes when depicting sets. LineSets [3] are the most minimal of all, reducing the geometry to a single continuous line per set which connects all elements. Both Kelp Diagrams [10] and its successor KelpFusion [15] are based on sparse spanning graphs, essentially variations of minimal spanning trees for different distance measures. These methods attempt to reduce visual clutter by reducing the amount of "ink" (see Tufte's rule [23]) necessary to connect all elements of a set. However, although the results are visually pleasing, neither method does use the optimal amount of ink. In this paper we explore the algorithmic questions that arise when computing spanning graphs for set visualization which are optimal with respect to ink usage.

Problem Statement. Our input is a set of n points in the plane. Each point is a member of one or more sets. We mostly consider the case where there are exactly two sets, namely a red and a blue set. A point is red if it is part only of the red set and it is blue if it is part only of the blue set. A point that is part of both the red and the blue set is purple.

A *red-blue-purple spanning graph* (RBP spanning graph) for a set of points that are red, blue and purple is a set of edges connecting 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. A *minimum RBP spanning graph* for a set of points that are red, blue and purple is a red-blue-purple spanning graph that has minimum weight (total edge length) (see Figure 1). In this paper we consider the algorithmic problems associated with computing minimum RBP spanning graphs.

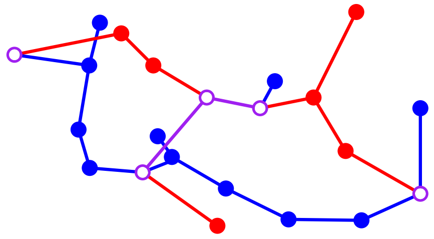


Fig. 1. A minimum RBP spanning graph

Results and Organization. We first review related work. In Section 2 we describe and prove various useful properties of (minimum) RBP spanning graphs. Then, in Section 3, we consider the two special cases where the points are located on a line or on a circle. This setting is meaningful if the elements of the sets are not associated with a specific location (for example, social networks or software systems). Here visualizations frequently choose to arrange elements in simple configurations such as lines or circles. We give an $O(n)$ time algorithm for points on a line, assuming that the input is already sorted. For points on a circle we exploit a structural result which allows us to use dynamic programming in $O(k^3 + n)$ time, where k is the number of purple points. In Section 4 we prove that computing a minimum RBP spanning graph is NP-hard in general. Hence, in Section 5 we turn to approximations. We describe an $O(n \log n)$ algorithm that computes a $(\frac{1}{2}\rho + 1)$-approximation of the minimum RBP spanning graph, where ρ is the Steiner ratio. Finally, in Section 6 we discuss various extensions for situations with more than two sets. Due to space constraints some proofs have been deferred to the full version.

Related Work. In recent years a number of papers explored the problem of automatically drawing Euler diagrams, for example, Simonetto and Auber [20], Stapleton *et al.* [21], and Henry Riche and Dwyer [13]. These methods assume that the locations of the set elements can be freely chosen. An important variation is the case that elements have to be drawn at fixed positions. Collins *et al.* [9] presented *Bubble Sets*, a method based on isocontours. A similar approach was suggested by Byelas and Telea [7]. *LineSets* by Alper *et al.* [3] attempt to improve the overall readability by the minimalist approach of drawing a single line per set. Dinkla *et al.* [10] introduced *Kelp Diagrams* which use a sparse spanning graph, essentially a minimum spanning tree with some additional edges. Kelp Diagrams were extended by Meulemans *et al.* [15] to a hybrid technique named *KelpFusion* which uses a mix of hulls and lines to generate fitted boundaries.

Sets defined over points in the plane can be interpreted as an embedding of a *hypergraph* where the points are vertices and each set is a hyperedge connecting an arbitrary number of vertices. Drawings of hypergraphs have been discussed in several papers (e.g., see Brandes *et al.* [5] and references therein).

Also in the area of discrete and computational geometry many problems on colored point sets have been studied. Possibly the most famous example is the *Ham-Sandwich Theorem*: given a set of $2n$ red points and $2m$ blue points in general position in the plane, there is always a line ℓ such that each open halfplane bounded by ℓ contains exactly n red points and m blue points. There have been many variations on this theorem and also many other results on finding configurations or geometric graphs with constraints depending on colors. We refer the interested reader to the survey [14] and to Chapter 8 in the collection of research problems [6].

From an algorithmic point of view, many problems have been considered, here we mention only a few of them. The *bichromatic closest pair* (e.g., see Preparata and Shamos [19] Section 5.7), the *chromatic nearest neighbor search* (see Mount et al. [18]), the problems on finding *smallest color-spanning objects* (see Abellanas *et al.* [1]), the *colored range searching* problems (see Agarwal *et al.* [2]), and the *group Steiner tree* problem where, for a graph with colored vertices, the objective is to find a minimum weight subtree that covers all colors (see Mitchell [16], Section 7.1). Finally, Tokunaga [22] considers a set of red or blue points in the plane and computes two geometric spanning trees of the blue and the red points such that they intersect in as few points as possible.

2 Properties of RBP Spanning Graphs

We call an edge of a RBP spanning graph *red* if it connects two red points or a red and a purple point. We call an edge *blue* if it connects two blue points or a blue and a purple point. We call an edge *purple* if it connects two purple points. A minimum RBP spanning graph does not contain edges between a red and a blue point. The subgraph induced by the red and purple points in a minimum RBP spanning graph is a spanning tree (and the analogous statement holds for the blue and purple points). Figure 2 (a) illustrates the trade-off between red, blue, and purple edges in a minimum RBP spanning graph.

Every red edge in a minimum RBP spanning graph also occurs in a minimum spanning tree of only the red and purple points. The corresponding statement is true also

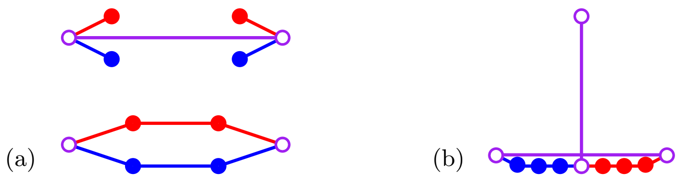


Fig. 2. (a) Two examples of minimum RBP spanning graphs of similar configurations of points. (b) A minimum RBP spanning graph with a purple edge crossing.

for the blue edges, but not for the purple edges. That is, there can be purple edges in a minimum RBP spanning graph which do not occur in a minimum spanning tree of only the purple points.

It is easy to see that a minimum RBP spanning graph is not necessarily planar. Red and blue edges are mostly independent and they can easily cross. Moreover, a red and a purple edge can cross, a blue and a purple edge can cross, and even two purple edges can cross in a minimum RBP spanning graph, as shown in Figure 2 (b). In fact, a single purple edge can cross arbitrarily many purple edges. The intricate construction and the additional observations necessary to prove this are relegated to the full version.

Lemma 1. *A purple edge in an optimal RBP spanning graph can cross $\Theta(n)$ other purple edges.*

Just as with standard minimum spanning trees the degree of the points in a minimum RBP spanning graph is bounded.

Lemma 2. *The maximum degree of a point in a minimum RBP spanning graph is at most 18.*

This bound can be attained by a purple point p: Let p be the center of a regular hexagon with radius 3 and two more regular hexagons with radius 1, one slightly rotated. Place a purple point on each corner of the large hexagon, place red points on the corners of one of the smaller hexagons, and blue points on the corners of the other one. Then the star graph with p at the center is a minimum RBP spanning graph. A similar construction shows that there is a point set such that the unique minimum RBP spanning graph requires a point of degree 15; a higher degree is never necessary. A red or blue point can have degree at most 6, degree 5 is the highest degree that can be enforced.

3 Points on a Line or on a Circle

Here we describe efficient algorithms to find a minimum RBP spanning graph if the points lie on a line or on a circle. In the circle case, we first present additional geometric observations which allow us to use dynamic programming.

3.1 Points on a Line

Given a problem instance S, we number the purple points $p_1, \ldots, p_k$ from left to right. For any $1 \leq i \leq k-1$, let S_i be the set of points between p_i and p_{i+1} (including both

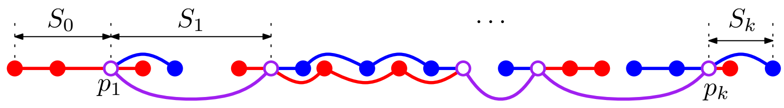


Fig. 3. A minimum RBP spanning graph of points on a line, and its partition into sets S_i (some edges are depicted by curved arcs for clarity).

p_i and p_{i+1}). We also define S_0 to contain p_1 and all red/blue points to its left, and S_k to contain p_k and all red/blue points to its right (see Figure 3). First, we show that each subset can be treated independently.

Lemma 3. *Let S be a set of red, blue and purple points located on a line, and let G^* be a minimum RBP spanning graph of S. Then for any edge $qq' \in G^*$, the points q and q' are contained in S_j, for some j.*

Using this lemma it is straightforward to obtain an efficient algorithm.

Theorem 1. *Let S be a set of n red, blue and purple points located on a line. We can compute a minimum RBP spanning graph of S in $O(n)$ time, provided that the points are sorted along the line.*

3.2 Points on a Circle

We proved in Lemma 1 that for points in general position a purple edge can cross many other purple edges. Even if the points lie in convex position, purple edges can cross each other (see Figure 2 (right)). But below we prove that for points on a circle the situation is structurally different and purple edges cannot cross any other edges.

Lemma 4. *Let S be a set of red, blue and purple points located on a circle. A minimum RBP spanning graph of S cannot have a purple edge crossing any other edge.*

Proof. Let G be a minimum RBP spanning graph in which two edges $e_1 = vv'$ and $e_2 = ww'$ cross. We will perform a local transformation that will reduce the weight of G, contradicting the minimality of G.

First assume that both e_1 and e_2 are purple, and consider the four red paths in G that start at either v or v' and end at w or w'. Without loss of generality, we can assume that the minimum-link path among the four (i.e., the path with smallest number of edges) connects v and w. Let π_R be such path; note that π_R cannot use edge vv' nor edge ww'. Likewise, let π_B be the minimum-link blue path among those that connect v or v' with w or w'. We now distinguish three cases depending on the number of shared endpoints between π_R and π_B (see Figure 4):

π_R and π_B share both endpoints. We replace edges vv' and ww' with edges vw' and $v'w$. The resulting graph G' clearly has smaller weight than G. We now prove that G' is indeed spanning. First consider the red tree in G': the removal of edge ww' created two components. Moreover, points v and w' must belong to different components (otherwise, the edge ww' would create a cycle in G). Thus, by adding

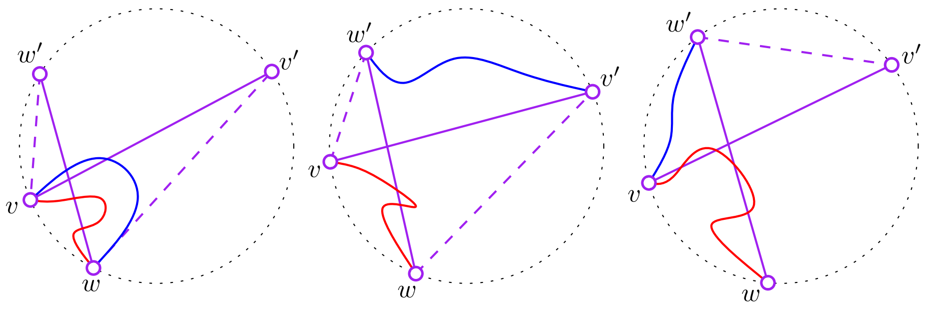


Fig. 4. The three cases in the proof of Lemma 4, local transformations are shown by dashed purple edges. No assumptions are made on the number of crossings between π_R, π_B, vv', and ww'.

the edge vw' we reconnect the two components again. Likewise, the removal of edge vv' creates two red components that are reconnected with the edge wv'. That is, graph G' also spans red. We repeat the same reasoning for blue and obtain that G' is a RBP spanning graph with smaller weight than G, a contradiction.

π_R **and** π_B **share no endpoints.** We proceed as in the previous case, replacing edges vv' and ww' by vw' and $v'w$. The argumentation is identical to the previous case.

π_R **and** π_B **share one endpoint.** We can assume that v is the shared endpoint, and that the other endpoint of π_B is w' (see Figure 4, right). In this case, both red and blue paths from v' to both w and w' in G use the edge vv'. Then we can replace this edge by either $v'w$ or $v'w'$ and maintain the spanning property. Using the fact that the four vertices are on the boundary of a circle, it is easy to see that either $||v'w|| < ||v'v||$ or $||v'w'|| < ||v'v||$ must hold, thus one of the two resulting graphs will have smaller weight.

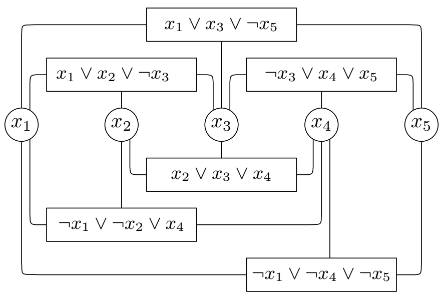
If one of the edges is not purple the situation is easier, since we need to consider only one color. We assume that the edge e_2 is red, and that the path from v to w does not use e_1 nor e_2. Then we can replace the edge ww' by either vw' or $v'w'$ to obtain a graph of smaller weight. Note that, since we are changing a red edge, the spanning property of blue cannot be altered, and the lemma is shown. □

Next we present another crossing property that will be useful for our algorithm.

Corollary 1. *Let S be a set of red, blue and purple points located on the boundary of a circle. In a minimum RBP spanning graph G of S, no red or blue edge of G can cross a segment between two purple points.*

Proof. Let p, p' be two purple points, and assume that a red edge $rr' \in G$ crosses the segment pp'. As in the proof of Lemma 4, we can assume that the red path from p to r does not pass through neither p' nor r'. Then, we replace the edge rr' by either $r'p$ or $r'p'$ to obtain a RBP spanning graph of smaller weight. □

Using these geometric observations and a few others proved in the full version we can now compute a minimum RBP spanning graph with dynamic programming.


$x_1 \lor x_3 \lor \neg x_5$
$x_1 \lor x_2 \lor \neg x_3$
$\neg x_3 \lor x_4 \lor x_5$
x_1
x_2
x_3
x_4
x_5
$x_2 \lor x_3 \lor x_4$
$\neg x_1 \lor \neg x_2 \lor x_4$
$\neg x_1 \lor \neg x_4 \lor \neg x_5$
(a)

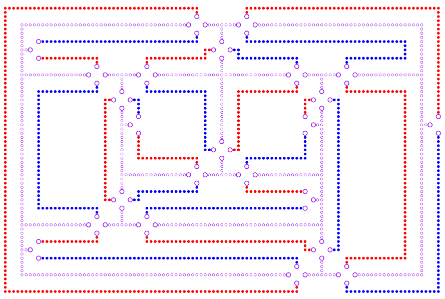


Fig. 5. (a) A planar 3-SAT formula. (b) The corresponding set of red, blue, and purple points.

Theorem 2. *Let S be a set of n red, blue and purple points located on a circle. We can compute a minimum RBP spanning graph of S in $O(k^3 + n)$ time, where k is the number of purple points.*

4 NP-hardness

Computing a minimum RBP spanning graph is NP-hard. We prove this by a reduction from planar 3-SAT. Figure 5 illustrates the global construction: an embedded 3-SAT formula, and the corresponding set of red, blue, and purple points that we construct. There is a value W such that a solution of weight less than W exists if and only if the 3-SAT formula is satisfiable.

The construction consists of two parts. First, we have a *variable gadget* for each variable x_i in the 3-SAT formula. Such a gadget consists of a purple *skeleton* and a red/blue *loop* around the skeleton. The loop consists of alternating densly-sampled blue and red chains, separated by a pair of purple points at distance $\sqrt{2}$ from each other. For each such pair of purple points there is also another purple point at distance 1 from both points, which is connected to the skeleton. We call such a group of 3 purple points a *switch*. Figure 6 (a) shows an example.

First, we observe that if the red, blue and purple chains are sampled sufficiently densly, then they will be connected in any optimal solution; hence, we ignore the costs of these connections from now on. There are exactly two ways to optimally connect the

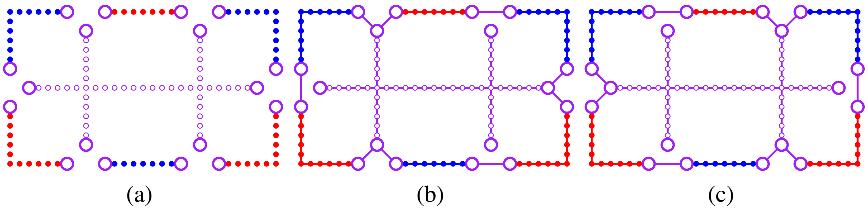


Fig. 6. (a) The input for a variable gadget. (b) One possible optimal solution of this construction, which represents the value *true*. (c) The other optimal solution represents the value *false*.

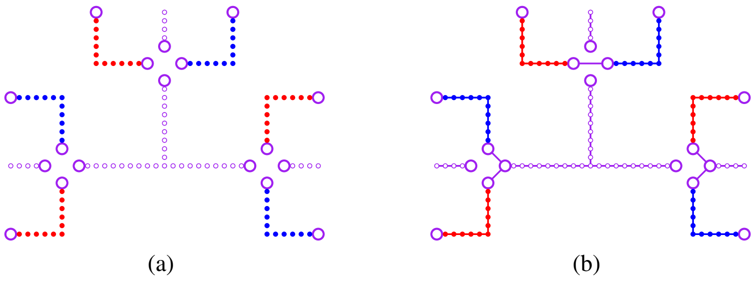


Fig. 7. (a) The input for a clause gadget. (b) One possible solution where the clause is satisfied. The left and top literals are in their *true* states; the right literal is in its *false* state.

remaining components: we alternatingly connect the purple points on opposite sides of a switch to each other, or both to the skeleton. Figures 6(b) and 6(c) illustrate the two solutions. These will correspond to the values *true* and *false* of the variable.

Next, we have *clause gadgets* for all clauses of the 3-SAT formula. These occur at places where the variable loops of the three involved variables get close to each other, and we make sure that there is a switch in each of them (from red to blue if the variable occurs positively in the clause, from blue to red otherwise). For these switches, we place a fourth point, also at distance 1 from both ends of the switch, and we connect these three extra points by a clause skeleton. Figure 7(a) shows an example.

Lemma 5. *There is a value W such that an RBP spanning graph of length less than W exists if and only if the 3-SAT formula is satisfiable.*

Proof. Suppose the planar 3-SAT formula has n variables and m clauses, and let k_i be the number of clauses that variable x_i appears in. There are $2k_i$ switches per variable. The total number of switches is $2\sum_{i=1}^{n} k_i = 6m$. We ignore the red, blue and purple chains; we only argue about the total length of purple edges within the switches.

Within variable x_i, there is one skeleton, k_i blue pieces, and k_i red pieces, which means that there are $3k_i + 1$ components in either the red or the blue tree that need to be connected. For this, we need to add $3k_i$ edges. Within each switch, the purple points that are connected to the red or blue paths must certainly get a purple edge. Since we add only $3k_i$ edges, half of the switches get only one edge; these edges must then connect the blue-path purple point to the red-path purple point and have length $\sqrt{2}$. The other half of the switches are connected via the skeleton with two edges of length 1. So, the total length within a variable is at least $k_i\sqrt{2} + 2k_i$. This is also possible to achieve, as seen in Figure 6(b) and 6(c).

Now, the clause skeletons have to be connected to at least one of the variables. For each of them, we need one more edge in one of the switches. The cheapest possible way of doing this is by using the groups that had expensive edges (of length $\sqrt{2}$) and using two normal edges (total length 2) instead. So, globally, of the $6m$ switches, at least $4m$ need two edges and the ones with a single edge must have length at least $\sqrt{2}$. Thus, the total length must be at least $W = (8 + \sqrt{2})m$. We claim that a solution of this length exists if and only if the formula is satisfiable. □

Theorem 3. *Computing a minimum RBP spanning graph is NP-hard.*

5 Approximation

A simple approximation algorithm determines the red-purple minimum spanning tree and the blue-purple minimum spanning tree, and takes the union of their edges. It is easy to see that this is a 2-approximation algorithm that requires $O(n \log n)$ time.

Another approximation algorithm, A, starts by computing the minimum spanning tree of the purple edges, and then adds the red and blue points in an optimal manner in the style of Kruskal's algorithm for minimum spanning trees. Algorithm A can also be implemented to run in $O(n \log n)$ time by computing the Delaunay triangulation of the red and purple points and of the blue and purple points. It is easy to argue that A also is a 2-approximation algorithm but interestingly, we can prove a better bound (close to 1.6) by expressing the approximation factor in the Steiner ratio ρ. Gilbert and Pollak [12] conjectured that $\rho = \frac{2}{\sqrt{3}} \approx 1.15$, but this conjecture has not been proved yet.[1] Chung and Graham [8] showed a bound of ≈ 1.21, which is the best-known upper bound on ρ.

Theorem 4. *Approximation algorithm A is a $(\frac{1}{2}\rho + 1)$-approximation of the minimum RBP spanning graph, where ρ is the Steiner ratio. The approximation is not a c-approximation for any constant $c < 1 + \frac{1}{\sqrt{3}}$.*

Proof. Algorithm A is not a c-approximation for any $c < 1 + \frac{1}{\sqrt{3}}$ (see Figure 8). Hence our approximation analysis is tight if the Gilbert-Pollak conjecture is true.

Next we prove our claim on the approximation factor. Let R, B, and P be sets of red, blue, and purple points. Let G^* be their minimum RBP spanning graph. Let R^* be the red edges, B^* the blue edges, and P^* the purple edges in G^*. Let A be the algorithm that computes a spanning graph by taking the minimum spanning tree of the purple points, and then adding the red and blue points optimally. We denote the resulting graph on $R \cup B \cup P$ by G', and its red, blue and purple edges by R', B', and P'.

Suppose first that G^* has no purple edges. Then algorithm A gives extra length in terms of purple edges equal to the MST of the purple points, denoted $||P'||$. The optimal

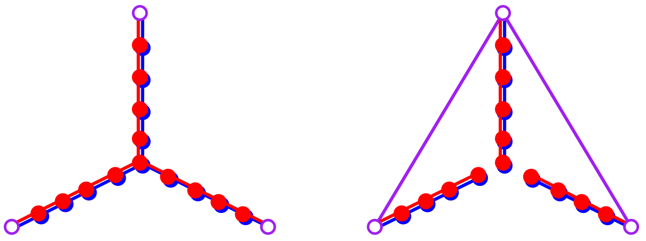


Fig. 8. A minimum RBP spanning graph and the RBP spanning graph on the same points obtained by approximation algorithm A

[1] A proof of the conjecture by Du and Hwang, "A proof of Gilbert-Pollak Conjecture on the Steiner ratio", *Algorithmica* 7:121–135 (1992), turned out to be incorrect.

graph G^* must connect all purple points simultaneously through a red spanning tree and through a blue spanning tree whose lengths are $||R^*||$ and $||B^*||$. Algorithm A has a total length of red edges of $||R'|| \leq ||R^*||$ and a total length of blue edges of $||B'|| \leq ||B^*||$. Hence the approximation ratio of A in case of absence of purple edges in the optimal solution is

$$\frac{||P'|| + ||R'|| + ||B'||}{||R^*|| + ||B^*||} \leq \frac{||P'|| + ||R^*|| + ||B^*||}{||R^*|| + ||B^*||} .$$

This ratio is maximized when R lies very densely on the Steiner Minimum Tree of P, and the same is true for B. Due to the density, algorithm A will choose nearly the full length of the Steiner Minimum Tree of P as well, once for red and once for blue. The approximation ratio is then smaller than but arbitrarily close to

$$\frac{MST(P) + 2 \cdot SMT(P)}{2 \cdot SMT(P)} \leq \frac{\rho \cdot SMT(P) + 2 \cdot SMT(P)}{2 \cdot SMT(P)} = \frac{\rho + 2}{2} = \frac{1}{2}\rho + 1 ,$$

where $SMT(P)$ is the Steiner Minimum Tree of P (or its length) and $MST(P)$ is the Minimum Spanning tree of P (or its length).

Next, suppose that G^* has a set P^* of purple edges, and assume them fixed. We will reason about sets of red, blue and purple points for which the algorithm A performs as poorly as possible in terms of approximation ratio.

If G^* has any red point r that has a single red edge incident to it in R^*, then this edge will connect r to the closest red or purple point, otherwise G^* is not optimal. Algorithm A will choose exactly the same edge in its solution. Hence, the approximation ratio of A for the points $R \setminus \{r\}$, B, and P is higher than for the points R, B, and P. The same is true for a blue or purple point that has a single incident edge in G^*. So we can restrict ourselves to analyzing point sets whose optimal solution does not have any leafs in G^*.

Let $B@R$ be a set of blue points infinitesimally close to the locations of the red points, and let $R@B$ be a set of red points infinitesimally close to the locations of the blue points. Now we can compare the approximation ratio of A (i) on P, R, and B, (ii) on P, R, and $B@R$, and (iii) on P, $R@B$, and B, and notice that at least one of (ii) and (iii) gives an approximation ratio at least as high as for (i). Hence, we can restrict ourselves to analyzing point sets where the red and blue points lie at basically the same positions (but they are not purple points).

The edges of P^* partition the purple points of P into a number of purple components which are connected by a red spanning forest and a blue spanning forest. We have

$$||P'|| \leq ||P^*|| + \rho||R^*|| ,$$

because in G^* the red (blue) connections between the purple components cannot be shorter than the Steiner Minimum Forest of the purple components. By the observations above we can assume that all red and blue points are used in the red and blue spanning forests that connect the purple components. Hence the approximation ratio is

$$\frac{||P'|| + ||R'|| + ||B'||}{||P^*|| + ||R^*|| + ||B^*||} \leq \frac{||P'|| + 2||R'||}{||P^*|| + 2||R^*||} \leq \frac{||P^*|| + \rho||R^*|| + 2||R^*||}{||P^*|| + 2||R^*||} .$$

This ratio is maximized when $||P^*|| = 0$, in which case we get exactly the same ratio as above, when no purple edges are present. □

It is possible that a PTAS exists for our problem, but it is not clear whether the techniques of Arora [4] or Mitchell [17] for the Euclidean traveling salesperson problem can be applied since RBP spanning graphs are not planar, and the number of crossings of a single edge can be large.

6 Extensions and Future Work

Beyond Purple. So far we considered the case where there are exactly two sets, the red set and the blue set, leading to an input with red, blue, and purple points. In general we might have k different sets, all denoted by primary colors. For instance, for $k = 3$ we could have red, blue and yellow sets, which leads to three secondary colors (purple, orange, green) and one tertiary color (black). The objective is again to minimize the total length of a multi-colored spanning graph which has the property that the subgraphs induced by the red, blue, and yellow sets are connected (see Figure 9 (a)).

This problem is clearly still NP-hard. The 2-approximation immediately generalizes to a 3-approximation (or a k-approximation for k primary colors). We can improve on this by incorporating our $(1 + \frac{1}{2}\rho)$-approximation algorithm to obtain a $(2 + \frac{1}{2}\rho)$-approximation for three sets, or more generally a $(\lceil\frac{1}{2}k\rceil + \lfloor\frac{1}{2}k\rfloor\frac{1}{2}\rho)$-approximation for k sets. Interestingly, our algorithms for points on a line or on a circle are not straightforward to generalize; these problems remain open.

Line Drawings. Another extension is motivated by LineSets [3]. Returning for the moment to the setting with two sets (and red, blue, and purple points), we now wish to compute a minimum RBP spanning graph such that the subgraphs induced by the red and blue sets are paths. That is, the red and purple edges form a path connecting all red and purple points, and the blue and purple edges form a path connecting all blue and purple points (see Figure 9 (b)).

This problem is NP-hard since TSP is hard. Nonetheless, we can obtain a $(2 + \varepsilon)$-approximation by independently computing an approximate TSP for the blue and purple points and for the red and purple points, and simply taking the union. An approach similar to the spanning tree case seems to fail and hence a better solution remains an open problem. The question also remains open for points on a line or on a circle.

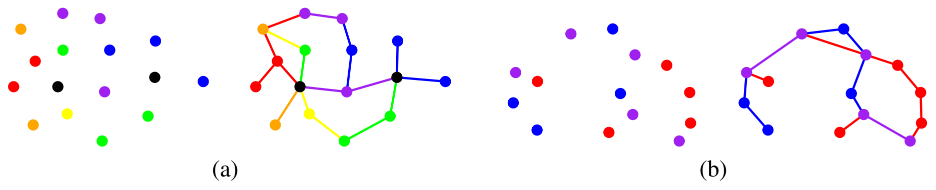


Fig. 9. (a) A set of multicolored points representing red, blue, and yellow sets and a corresponding spanning graph. (b) A set of red, blue and purple points, and a graph that connects all red points in a path and all blue points in a path.

References

1. Abellanas, M., Hurtado, F., Icking, C., Klein, R., Langetepe, E., Ma, L., Palop, B., Sacristán, V.: Smallest color-spanning objects. In: Meyer auf der Heide, F. (ed.) ESA 2001. LNCS, vol. 2161, pp. 278–289. Springer, Heidelberg (2001)
2. Agarwal, P.K., Govindarajan, S., Muthukrishnan, S.M.: Range searching in categorical data: Colored range searching on grid. In: Möhring, R.H., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, pp. 17–28. Springer, Heidelberg (2002)
3. Alper, B., Riche, N., Ramos, G., Czerwinski, M.: Design study of LineSets, a novel set visualization technique. IEEE TVCG 17(12), 2259–2267 (2011)
4. Arora, S.: Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. J. ACM 45(5), 753–782 (1998)
5. Brandes, U., Cornelsen, S., Pampel, B., Sallaberry, A.: Path-based supports for hypergraphs. J. Discrete Algorithms 14, 248–261 (2012)
6. Brass, P., Moser, W.O.J., Pach, J.: Research Problems in Discrete Geometry. Springer, New York (2005)
7. Byelas, H., Telea, A.: Towards realism in drawing areas of interest on architecture diagrams. Visual Languages and Computing 20(2), 110–128 (2009)
8. Chung, F., Graham, R.: A new bound for Euclidean Steiner minimum trees. Ann. N.Y. Acad. Sci. 440, 328–346 (1986)
9. Collins, C., Penn, G., Carpendale, S.: Bubble Sets: Revealing set relations with isocontours over existing visualizations. IEEE TVCG 15(6), 1009–1016 (2009)
10. Dinkla, K., van Kreveld, M., Speckmann, B., Westenberg, M.A.: Kelp Diagrams: Point set membership visualization. Computer Graphics Forum 31(3), 875–884 (2012)
11. Edwards, A.W.F.: Cogwheels of the mind. John Hopkins University Press (2004)
12. Gilbert, E., Pollak, H.: Steiner minimal trees. SIAM J. Appl. Math. 16, 1–29 (1968)
13. Henry Riche, N., Dwyer, T.: Untangling Euler diagrams. IEEE TVCG 16(6), 1090–1099 (2010)
14. Kaneko, A., Kano, M.: Discrete geometry on red and blue points in the plane – a survey. In: Discrete and Comp. Geometry, The Goodman-Pollack Festschrift, pp. 551–570 (2003)
15. Meulemans, W., Henry Riche, N., Speckmann, B., Alper, B., Dwyer, T.: KelpFusion: a hybrid set visualization technique. In: IEEE TVCG (to appear, 2013)
16. Mitchell, J.S.B.: Geometric shortest paths and network optimization. In: Handbook of Computational Geometry, pp. 633–701 (1998)
17. Mitchell, J.S.B.: Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. SIAM J. Comput. 28(4), 1298–1309 (1999)
18. Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.Y.: Chromatic nearest neighbor searching: A query sensitive approach. Computational Geometry: Theory and Applications 17(3-4), 97–119 (2000)
19. Preparata, F.P., Shamos, M.I.: Computational Geometry: An Introduction. Springer, New York (1985)
20. Simonetto, P., Auber, D.: Visualise undrawable Euler diagrams. In: Proc. 12th Conf. on Information Visualisation, pp. 594–599 (2008)
21. Stapleton, G., Rodgers, P., Howse, J., Zhang, L.: Inductively generating Euler diagrams. IEEE TVCG 17(1), 88–100 (2011)
22. Tokunaga, S.: Intersection number of two connected geometric graphs. Information Processing Letters 59(6), 331–333 (1996)
23. Tufte, E.R.: The Visual Display of Quantitative Information. Graphics Press (1983)