Computational Geometry 58 (2016) 124–135

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

Alternating paths and cycles of minimum length ☆

W. Evans [a], G. Liotta [b], H. Meijer [c], S. Wismath [d,*]

[a] University of British Columbia, Canada
[b] Universitá degli Studi di Perugua, Italy
[c] U. C. Roosevelt, The Netherlands
[d] University of Lethbridge, Canada

ARTICLE INFO

Article history:
Received 23 December 2015
Received in revised form 14 April 2016
Accepted 18 July 2016
Available online 21 July 2016

Keywords:
Alternating paths/cycles
Colored points

ABSTRACT

Let R be a set of n red points and B be a set of n blue points in the Euclidean plane. We study the problem of computing a planar drawing of a cycle of minimum length that contains vertices at points $R \cup B$ and alternates colors. When these points are collinear, we describe a $\Theta(n \log n)$-time algorithm to find such a shortest alternating cycle where every edge has at most two bends. We extend our approach to compute shortest alternating paths in $O(n^2)$ time with two bends per edge and to compute shortest alternating cycles on 3-colored point sets in $O(n^2)$ time with $O(n)$ bends per edge. We also prove that for arbitrary k-colored point sets, the problem of computing an alternating shortest cycle is NP-hard, where k is any positive integer constant.

1. Introduction

A recent paper by Chan et al. [5] studies the problem of computing a planar drawing of an n-vertex planar graph such that the vertex locations are given as part of the input and the drawing has minimum total edge length. The problem is known to be NP-hard [4] in general and Chan et al. describe different polynomial time approximation algorithms for paths, matchings, and general planar graphs. They also give a polynomial time exact algorithm for paths on fixed positions that lie on a line, which computes a planar drawing where all edges are monotone in a common direction and each edge can be represented by a poly-line having $O(n)$ bends.

In this paper we consider a variant of the problem by Chan et al. where the position for each vertex is not fixed, but it can be chosen by the algorithm as one in a given subset of a point set. To be precise, we are given a k-colored graph (i.e., a graph where each vertex has one of k different colors) and we want to compute a planar drawing of the graph on a given k-colored point set so that vertices are mapped to distinct points of the same color and the total edge length is minimized. We assume that for each color class the cardinalities of the sets of vertices and points match.

We mainly focus on drawing shortest alternating 2-colored (bicolored) paths and cycles on collinear point sets. But we also consider the case of more than two colors and the case that the points are non-collinear. Our main results are:

☆ The research reported in this paper started at the 2015 Bertinoro Workshop, sponsored by the EuroGIGA Project. Research also supported by NSERC, and by MIUR of Italy under project AlgoDEEP prot. 2008TFBWL4. An early version of this work appeared at the International Symposium on Graph Drawing and Network Visualization GD 2015.

* Corresponding author.
E-mail address: wismath@uleth.ca (S. Wismath).

http://dx.doi.org/10.1016/j.comgeo.2016.07.005
0925-7721/

- Let R be a set of n red points and B be a set of n blue points such that all points are distinct and collinear. We describe a $\Theta(n \log n)$-time sweep-line algorithm to compute a planar drawing of an alternating cycle (of $2n$ vertices) of minimum length on $R \cup B$ such that every edge is a poly-line with at most two bends.
- We adapt the approach for cycles to the problem of computing a shortest alternating path on a bicolored set of collinear points. We describe an $O(n^2)$-time algorithm that solves the problem by computing drawings with at most two bends per edge.
- We extend the study to 3-colored collinear point sets and describe an $O(n^2)$-time algorithm to compute shortest alternating cycles (visiting the colors in cyclic order) such that every edge has $O(n)$ bends.
- We consider non-collinear point sets and prove that computing a shortest alternating cycle is NP-hard in the general case of k-colored point sets, where $k \geq 1$ is a given constant.

From a technical point of view, our drawing algorithms are based on the idea of computing an alternating topological book embedding of a path or cycle such that the number of edges that are intersected by any cut is minimum. This approach seems to be specific for two and three colors, since we also present an example with four colors where an alternating cycle of minimum length cannot match the cut lower bound that we use for fewer than four colors.

1.1. Related work

The problem of computing a planar alternating path or a planar alternating cycle on $R \cup B$ has a long tradition in graph drawing and computational geometry. While the interested reader may refer to the survey by Kaneko and Kano [15] for a list of early references, we briefly recall here some of the milestone results.

Akiyama and Urrutia [2] study straight-line alternating paths when $R \cup B$ is in convex position; they exhibit a set of sixteen points for which a straight-line alternating path does not exist and present an $O(n^2)$-time algorithm to test when a straight-line alternating path on points in convex position exists. Abellanas et al. [1] show that if either the convex hull of $R \cup B$ consists of all the red points and no blue points or there exists a line that separates all blue points from red ones, then a straight-line alternating path always exists. Kaneko, Kano, and Suzuki [17] characterize those point sets in general position for which a straight-line alternating path always exists: If $R \cup B$ consists of at most twelve points or if it consists of exactly fourteen points, then a straight-line alternating path always exists; for all other cases, there exist configurations of red and blue points for which a straight-line alternating path does not exist. These early results about straight-line alternating paths have motivated further research on computing alternating paths and cycles when the edges can bend. Di Giacomo et al. [9] prove that every point set admits an alternating path and an alternating cycle with at most one bend per edge. The technique of Di Giacomo et al. [9] is based on projecting the points on a horizontal line and then computing a book embedding on this line before mapping the edges back to the original points. The main differences between the technique of [9] and the one described in this paper is that we shall compute topological book embeddings (i.e. we allow spine crossings which is not the case in [9]) and that we choose the vertices so to minimize the path/cycle length at the expense of having two bends per edge instead of just one.

The above mentioned studies on alternating paths/cycles have been extended by several authors who study either the case that the vertices of the cycle (path) are colored with more than two colors, or the case that the input is a bicolored graph more complex than a cycle (or path), or the combination of these two conditions (i.e. the input is a k-colored graph for $k > 2$). In some of these papers the edges are required to be drawn as straight-line segments, in some others adding bends along the edges is allowed. Also, in these generalizations the input graph may not be properly colored, that is adjacent vertices may be given the same color. We use the next two paragraphs to briefly recall some of the most relevant papers on these problems; see also Section 8.5.2 of [6] for additional references.

For bicolored graph families other than paths or cycles, the input is a bicolored planar graph G together with a bicolored set of points in the plane and the goal is to compute a planar drawing of G such that every red vertex is mapped to a red point and every blue vertex is mapped to a blue point and either each edge is a straight-line segment or it has a constant number of bends. Frati et al. [12] prove that it is NP-hard deciding whether a bicolored tree admits a straight-line drawing on a bicolored set of points, while van Garderen et al. [21] present linear time solutions restricted to special types of bicolored trees (e.g. those whose leaves all have the same color) and to special types of point sets. Special types of bicolored trees are also considered by Kaneko and Kano [13] who show that a forest consisting of at most two trees each having exactly one red vertex always admits a straight-line drawing on any bicolored set of points. Kaneko and Kano extend this result to forests with more than two trees and with exactly one red vertex per tree, under the assumption that the forest consists of either star-trees [14] or that it is formed by trees whose sizes differ from one another by at most one vertex [16]. Di Giacomo et al. [8] show that any forest with exactly one red vertex per tree admits a planar straight-line drawing on any bicolored point set in convex position. Concerning graphs other than trees, it is known that every bicolored outerplanar graph admits a planar drawing with at most five bends per edge on any bicolored point set [7], but there are bicolored triangulations and bicolored point sets for which any planar drawing requires linearly many bends on at least one edge [9].

For more than two colors, the input consists of a planar graph G whose vertices are partitioned with k distinct colors and of a set of points S also partitioned with the same k colors and such that for each color c the number of points of S having color c equals the number of vertices of G having color c. The desired output is a planar drawing of G such that

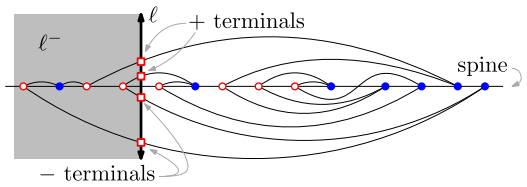

Fig. 1. An alternating cycle cut by a sweep-line ℓ.

every vertex of color c is mapped to a distinct point of S having color c. In the extreme case, one is given n colors modeled as n numbers from 1 to n and the goal is to compute a planar drawing where the location of the vertices is specified as part of the input. A seminal result in this context is due to Pach and Wenger [19] who prove that linearly many bends per edge are always sufficient and sometimes necessary for n-colored paths (cycles) and n-colored point sets in convex position. Their drawing technique can be extended to general n-colored planar graphs and the number of bends per edge was improved by Badent et al. [3]. The case when the number of colors is a given constant k has also been studied. Estrella-Balderrama et al. [11] describe special 3-colored points sets for which families of 3-colored trees admit a straight-line drawing [10] and Di Giacomo et al. [10] show that a k-colored planar graph can be drawn with constant number of bends per edge onto any k-colored set of points such that for each color there is a vertical strip that separates all points of that color from all other points. The study of families of k-colored graphs that admit a planar drawing on k-colored point sets with constant number of bends per edge remains a largely unexplored research direction.

1.2. Paper organization

The remainder of the paper is organized as follows. An overview of our algorithmic approach is presented in Section 2. Section 3 describes the algorithm for shortest alternating cycles on collinear red–blue points. Shortest alternating paths on collinear red–blue points are studied in Section 4. We consider shortest alternating cycles on collinear points with more than two colors in Section 5. In Section 6, we show that finding the shortest alternating cycle for non-collinear points is NP-hard. Finally, open problems are listed in Section 7.

2. Overview of the algorithmic approach

Let R be a set of n red points and B be a set of n blue points such that all points are distinct and in the Euclidean plane. An *alternating cycle* (*alternating path*) is a drawing of a cycle (path) such that the vertex set of the drawing is the set $R \cup B$ and such that no two vertices having the same color are adjacent. The drawing is *planar* if no two edges cross. The length of the cycle (path) is the sum of the lengths of its edges. A *shortest* alternating cycle (path) is one of minimum length. In this paper we are interested in computing shortest planar alternating cycles (paths). Since the problem is NP-hard for general point sets (Section 6), we focus on collinear point sets and assume that the line through the point set, called the *spine*, is horizontal.

For collinear point sets, the length of an edge, which may have bends, is defined to be the (Euclidean) distance between its endpoints in the point set, rather than the length of the curve representing the edge. We are interested in solving the combinatorial problem of finding an order in which a shortest alternating cycle (path) visits the colored points, and the planar embedding of its edges. Once this is found, a planar alternating cycle (path) approaching this length can be obtained by making the edge curves "as flat as possible" around the spine, that is by making the distance between each edge and the spine tend to zero. Hence when we say that we "compute the shortest cycle (path)", we mean that we compute an ordering and embedding for which such a cycle (path) exists.

A set of n blue points and n red points on a line define $2n + 1$ disjoint intervals, two of which are infinite. Assume we have a (not necessarily optimal) planar drawing of an alternating cycle (path). Consider a vertical line in any interval and count the number of edges of the cycle (path) intersected by the line. If we multiply the length of each finite interval by the number of edges that are intersected by a vertical line through the interval and then sum up all the obtained numbers, we obtain a lower bound on the length of the cycle. Therefore, we aim at computing an alternating cycle (path) C such that for any vertical line ℓ, the number of edges of C cut by ℓ is the minimum over all alternating cycles (paths). See Fig. 1. In addition, no two edges of C cross and every edge is a poly-line consisting of at most three segments (i.e. it has at most two bends). For brevity, in what follows we will often say alternating cycle (path) to mean planar alternating cycle (path).

Based on the observation above, the problem turns into the computation of a special type of topological book embedding, such that every edge can cross the spine at most once and such that the number of edges that span any interval between two consecutive points along the spine is minimum. Every edge of such a topological book embedding can be represented as a poly-line with at most two bends. Recall that a topological book embedding is a planar drawing of a graph such that all vertices are points of a line called the *spine* and the edges are simple Jordan arcs.

3. Shortest alternating cycle on collinear red–blue points

Following the approach of Section 2, we start by establishing a lower bound on the number of edges of any alternating cycle intersected by a vertical line. Next, we present a sweep-line algorithm to compute a topological book embedding such that every interval is spanned by the minimum number of edges and such that every edge crosses the spine at most once.

3.1. A lower bound lemma

The following lemma establishes the lower bound that will be used to prove the optimality of the alternating cycles.

Lemma 1. *Let R be a set of n red points and B be a set of n blue points such that all points are distinct and on the x-axis. Let ℓ be a vertical line that intersects the x-axis between two points of $R \cup B$. If there are r red points and b blue points to the left of ℓ, then any alternating cycle on $R \cup B$ crosses ℓ at least $2\max\{1, |r-b|\}$ times.*

Proof. Let C be an alternating cycle on $R \cup B$ and ℓ^- be the halfplane to the left of ℓ. In each component of $C \cap \ell^-$, the number of points of one color can be at most one more than the number of points of another color. Thus, the minimum number of components of C to the left of ℓ is $|r-b|$. If the line ℓ lies between two vertices of C (i.e. it is not to the left of the leftmost vertex of C and it is not to the right of the rightmost vertex of C), then the number of components to the left of ℓ is also at least one, and the number of edges of C that intersect ℓ is twice the number of components in $C \cap \ell^-$. □

3.2. A sweep-line algorithm

We now describe a sweep-line algorithm that computes the shortest alternating cycle of a set of n red points and n blue points lying on the horizontal line $y = 0$, called the *spine*. We call our algorithm `Spine-Sweep`.

`Spine-Sweep` first orders the points by increasing x-coordinate and then it sweeps a vertical cut line ℓ across the points. The algorithm maintains a set of disjoint curves to the left of ℓ, each of which has both endpoints, called *terminals*, on ℓ. These curves are the connected components of the intersection of some shortest alternating cycle with the halfplane, ℓ^-, to the left of ℓ. The terminals are colored red or blue depending on the color of the closest colored point on the curve. Terminals above the spine are *positive* and those below are *negative*; this is called the *sign of the terminal*. See Fig. 1. If both terminals of a component have the same sign, then this is the *sign of the component*, otherwise the component *straddles* the spine. The *distance* of a component to the spine is the minimum number of terminals between one of its terminals and the spine. Two terminals are *adjacent* if the segment connecting them contains no other terminals. Note that these definitions are with respect to the current sweep-line ℓ; the distance of a component to the spine, for example, may change as the line ℓ moves.

During the sweep, components are created or merged when ℓ encounters a colored point. By carefully selecting which components to create and merge and how to merge them, the algorithm maintains the following invariants:

P1. If there is exactly one component and its terminals have different colors, then its terminals have different signs.
P2. If there are more than two terminals, then they all have the same color.
P3. The two closest components to the spine do not have the same sign.

When the algorithm encounters a colored point, p, it either *forks* a new component, if p's color is the same as the color of all terminals (or there are no terminals), otherwise it *merges* p with one or two existing components creating a single new component. We describe these two cases under the assumption that the encountered point p is blue. Symmetric operations hold if p is red. In the next two figures, the terminals are drawn as squares; also, light/dark vertices are red/blue.

Fork: If there are no red terminals, we create a new component containing (blue) p that straddles the spine and has adjacent (blue) terminals. See Fig. 2(a). While the figure shows an existing blue component with a positive and a negative terminal, we perform a fork operation in *all* cases when there are no existing red terminals.

Merge: If there are red terminals, we create a new component that contains (blue) p. If there is only one component, we add p to that component by extending the edge from the closest red terminal to p. If p is not the last colored point, we add a new edge from p to a new (blue) terminal so that the new component straddles the spine. If p is the last colored point, we extend the edge from the other (red) terminal to p. See Fig. 2.

If there are at least two components, then by property P2 all their terminals are red. Let K and J be the two closest components to the spine. We extend the edges from a terminal from K and a terminal from J to p. We choose the terminals and route the edges from all terminals to ensure that our invariant properties remain true.

By property P3 and the fact that components do not intersect, the configuration of the two closest components to the spine is one of the four shown schematically in the top row of Fig. 3. For two of these configurations, (c) and (d) in Fig. 3, we extend the edges from the closest terminals to the spine from K and J to p, and extend all other terminals horizontally. In the other two configurations, we consider the sign of the two closest components

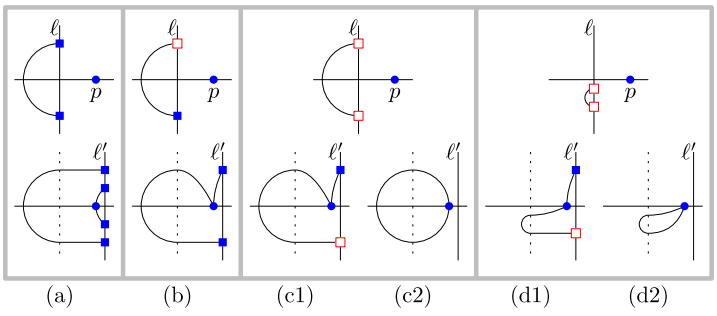

Fig. 2. The top row shows the initial configurations (the symmetric versions of (b) and (d) are not shown). The bottom row shows the possible configurations after merging. (a) Fork. One component is shown in the initial configuration but there may be many or none as long as all terminals are blue. The two new terminals are closest to the spine. (b)–(d) Merge with one component. All components are shown. Cases (c2) and (d2) only occur when p is the last colored point.

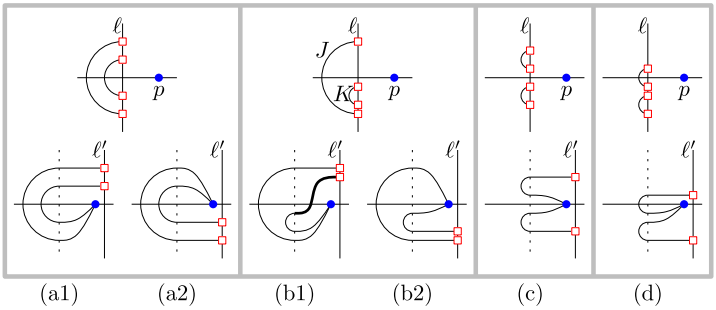

Fig. 3. Merge with two components. The top row shows the four basic configurations (the symmetric versions of (b) and (d) are not shown). The bottom row shows the possible configurations after merging.

to the spine *after* the merge. One of these components is the new component. If the other component exists and is negative, we merge to form configuration (a1) or (b1) in Fig. 3. Otherwise, we merge to form configuration (a2) or (b2) in Fig. 3. The result is that the two closest components to the spine after the merge have opposite signs. This preserves property P3.

Notice that forming configuration (b1) in Fig. 3 causes an edge of the alternating cycle (shown in bold) to cross the spine. We ensure that this edge is not forced to cross the spine again, which implies that each edge of the alternating cycle produced by the algorithm can be drawn with at most two bends. See also Lemma 5.

3.3. Main theorem

We prove that `Spine-Sweep` computes an alternating cycle C such that each edge crosses the spine at most once and no two edges cross each other. Also, any vertical line ℓ that intersects C, does so exactly $2\max\{1, |r-b|\}$ times, where r and b denote the number of red and blue points to the left of ℓ. Therefore, by using Lemma 1 and the observations that: (i) each fork/merge operation can be executed in constant time (for example by using a stack to maintain the components sorted according to their distance from the spine); and (ii) the red and blue points must be sorted in increasing x-order, we obtain the following:

Theorem 2. *Let R be a set of n red points and B be a set of n blue points such that all points are distinct and collinear. There exists an optimal $\Theta(n \log n)$-time algorithm that computes a planar alternating cycle of minimum length with at most two bends per edge.*

We prove the correctness of `Spine-Sweep` and our main result about computing shortest alternating cycles on collinear bicolored point sets via the following sequence of lemmas.

Lemma 3. `Spine-Sweep` *maintains properties P1–P3.*

Proof. The algorithm only produces a component with different colored terminals when there is a single component whose terminal colors differ from the color of p. See cases (c1) and (d1) in Fig. 2. Thus property P1 holds.

Property P2 is true after the sweep-line encounters the first colored point (causing a fork), and remains true following the fork and merge rules. Notice that the color of the terminals is the majority color of colored points to the left of sweep-line ℓ.

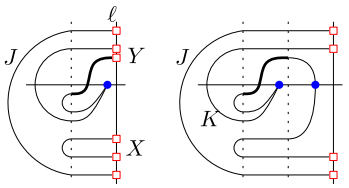

Fig. 4. Left: When e initially crossed the spine. Right: A future sweep-line when the component containing e could create an edge that crosses the spine. It could not be the same edge e again.

If there is no majority, there is one component with different colored terminals, or if all colored points are to the left of ℓ, there is one component to the left of ℓ that forms a cycle and doesn't intersect ℓ.

Property P3 is preserved by the rule that chooses which of cases (a1) or (a2) and cases (b1) or (b2) from Fig. 3 to apply when a merge occurs. □

Lemma 4. `Spine-Sweep` *computes an alternating cycle C. Also, let ℓ be a vertical line that intersects C. Line ℓ intersects C exactly $2\max\{1, |r-b|\}$ times, where r and b denote the number of red and blue points to the left of ℓ.*

Proof. We claim that the number of components to the left of ℓ is $\max\{1, |r-b|\}$, or 0 if $r = b = 0$. Also the color of all terminals is red if $r > b$, blue if $b > r$, and if $r = b > 0$ there is exactly one component whose terminals have different colors. The claim is certainly true if $r = b = 0$. If there is only one colored point to the left of ℓ, there is only one component whose terminals are the same color as the point (see Fig. 2(a)), and the claim is true. Suppose the claim is true when there are $k = r + b$ points to the left of ℓ. Consider the next point $p \in R \cup B$ encountered by the sweep-line, and assume p is blue. Call the line at this new position ℓ' with r' and b' the number of red and blue points to its left. If $r = b$, there is exactly one component at ℓ and it has different colored terminals. Since p is blue, $r' = r$, $b' = b + 1$, and $|r' - b'| = 1$ at ℓ'. The algorithm produces one component at ℓ' with two blue terminals (see Fig. 2(b)), and the claim is true. If $r - b = 1$, there is one component at ℓ and it has two red terminals. Since p is blue, the algorithm produces one component at ℓ' with one red and one blue terminal (see Fig. 2(c1 & d1)) or no terminals if p is the rightmost point in $R \cup B$ (see Fig. 2(c2 & d2)). In both cases, the claim is true. If $r - b > 1$, the algorithm merges two components with red terminals reducing the number of components by one, and the claim is true. If $b - r \geq 1$, the algorithm forks a new component increasing the number of components by one, and the claim is true. This exhausts the possible cases and establishes the claim for $k + 1$ points to the left of the sweep-line. A symmetric argument establishes the inductive step when p is red. Thus the claim is proved.

It follows that before the last point is encountered, we have one component whose terminals are both red if the last point is blue, and both blue otherwise. We connect both terminals with this last point (see Fig. 2(c2 & d2)), which closes the cycle. The merge rules ensure that the cycle is alternating.

Also from the claim, we know that the number of edges of the cycle cut by ℓ, which equals twice the number of components to the left of ℓ, matches the lower bound described in the statement of Lemma 1. □

Lemma 5. `Spine-Sweep` *computes an alternating cycle such that each edge crosses the spine at most once and no two edges cross each other.*

Proof. We observe that `Spine-Sweep` makes an edge cross the spine only when performing a merge operation involving two components as in Fig. 3(b1). Consider one such merge operations and let e be the edge that crosses the spine. We show that e is never extended across the spine again.

It suffices to show that no future sweep-line intersects e to create the terminal of the signed component, K, in Fig. 3(b) that is closer to the spine (since this is the only case when an edge is extended across the spine). Suppose some sweep-line ℓ does, and let K be the signed component, with e creating its closest terminal to the spine, and J the straddling component in Fig. 3(b). Since any newly created straddling component has adjacent terminals, component J could not be created after edge e initially crossed the spine. Therefore, component J has remained unaltered (except for horizontally extending its terminal edges) since e initially crossed the spine. When e initially crossed the spine, both a positive and negative component (X and Y in Fig. 4) were closer to the spine than J since the algorithm chose to create configuration (b1) in Fig. 3. (We may assume without loss of generality that point p is blue and X is a negative component.) These two components, plus whatever components are created by fork operations, must be merged into the single component K in order for J to be the straddling component in Fig. 3(b). Thus, at some time prior to sweep-line ℓ, the component containing e must have been merged with another component. Since e is the closest terminal to the spine in that component, such a merge would cause edge e to end, as in case (c) or (d) in Fig. 3. Thus the algorithm does not cause an edge e to cross the spine twice.

To see that no two edges cross one can use induction on the number of colored points to the left of ℓ. If only one point is to the left, its two adjacent edges do not cross initially. If after sweeping across k colored points no two edges cross, observe that neither the merge nor the fork operation makes two edges cross. □

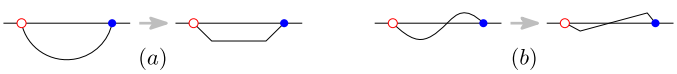

Fig. 5. Illustration for the proof of Theorem 2. Drawing an edge of an alternating cycle with at most two bends.

Lemmas 4 and 5 can be used to prove the main theorem.

Proof of Theorem 2. By Lemma 4, `Spine-Sweep` computes a planar alternating cycle C. Note that `Spine-Sweep` can be executed in $O(n \log n)$ time: After sorting the points, each merge/fork operation can be performed in constant time by maintaining a reference to the two terminals that are closest to the spine. Again by Lemma 4, for any vertical line intersecting C the number of intersected edges is the minimum. Hence by drawing every edge arbitrarily close to the spine, we obtain a cycle of minimum length. Let (u, v) be an edge of C. By Lemma 5, (u, v) crosses the spine at most once. If (u, v) does not cross the spine, it is drawn with two bends as illustrated in Fig. 5(a). If e crosses the spine, it is drawn with two bends as illustrated in Fig. 5(b).

Any algorithm that finds a shortest alternating cycle for distinct collinear points can be used to sort distinct integers $x_1, x_2, \ldots, x_n$ by creating a red point at $(x_i, 0)$ and a blue point at $(x_i + \epsilon, 0)$ for all i. If ϵ is smaller than one, the shortest alternating cycle has the points occur in sorted order. □

Note that the time complexity of Theorem 2 is worst-case optimal. Namely, if the red and blue points alternate along the spine, computing a shortest alternating cycle is equivalent to computing a circular sorting of the point set.

4. Shortest alternating paths on collinear red–blue points

We can also obtain a shortest alternating path on a set of n red and n blue points that are all distinct and collinear, provided the endpoints of the path are specified. The approach is the same as in the cycle case: we prove a lower bound on the number of times any alternating path with these endpoints intersects a vertical line ℓ and use essentially the same algorithm to find an alternating path that matches the bound. The lower bound is complicated slightly by the path endpoints.

Lemma 6. *Let R be a set of n red points and B be a set of n blue points such that all points are distinct and on the x-axis. Let ℓ be a vertical line that intersects the x-axis between two points of $R \cup B$. If there are r red points and b blue points to the left of ℓ, then the number of times any alternating path on $R \cup B$ with endpoints $u \in R$ and $v \in B$ crosses ℓ is at least:*

$$\begin{aligned} &2\max\{1, |r-b|\} && \text{if } u \text{ and } v \text{ are on the same side of } \ell,\\ &1 + 2\max\{b-r, r-b-1\} && \text{if only } u \text{ is left of } \ell,\\ &1 + 2\max\{r-b, b-r-1\} && \text{if only } v \text{ is left of } \ell. \end{aligned}$$

Proof. If both path endpoints u and v are on the same side of ℓ, the proof is the same as in the cycle case. If only the red path endpoint u is left of ℓ, then its component can have at most one more red point than blue points, and at most zero more blue points than red points. Thus, if $r > b$ this component can account for one of the excess $r - b$ red points, while if $b \geq r$, it cannot account for any of the excess $b - r$ blue points. This component crosses ℓ once; all others cross twice. If only the blue path endpoint v is left of ℓ, a symmetric argument applies. □

To find a shortest alternating path between two given endpoints, we can use a modification of `Spine-Sweep`. More precisely, a fork operation on a vertex of degree one gives rise to a component with only one terminal and a merge operation on a vertex of degree one joins the closest terminal of the closest component. The main difference with the approach described in the previous section is that we may have an odd number of terminals during some steps of the sweep-line procedure, which however does not change the reasoning behind either the proof of correctness or the time complexity.

Lemma 7. *Given a set R of n red points and a set B of n blue points such that all points are distinct and collinear, and given $u \in R$ and $v \in B$, there exists a $\Theta(n \log n)$-time algorithm that computes a planar alternating path of minimum length with at most two bends per edge that starts at u and ends at v.*

Suppose we want to find the shortest alternating path but it may start and end at any pair of points. While one might think that a shortest alternating path always starts at the leftmost red (blue) point and always ends at the rightmost blue (red) point, this is not always the case. For example, the point set of Fig. 6 has an alternating path in Fig. 6(a) whose length is minimal if it is required that both endpoints are extremal but it is not as short as the one in Fig. 6(b).

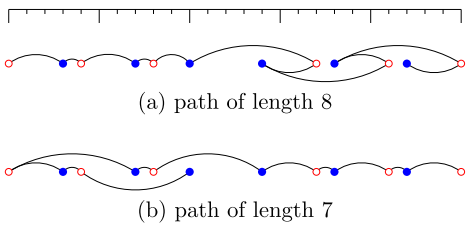

Fig. 6. Alternating paths.

To find the best endpoints, we may use the fact that our algorithm matches the lower bound described in Lemma 6. Let r_i and b_i be the number of red and blue points, respectively, among the first (leftmost) i colored points in $R \cup B$. (Note: $r_i + b_i = i$.) Let $c_i = 2\max\{1, |r_i - b_i|\}$, $s_i = 1 + 2\max\{b_i - r_i, r_i - b_i - 1\}$, and $t_i = 1 + 2\max\{r_i - b_i, b_i - r_i - 1\}$. Let d_i be the distance between the ith and $(i+1)$st colored points. If the path starts at the jth and ends at the kth colored point, then its minimum length is the sum of the lower bounds from Lemma 6 weighted by the distance between adjacent colored points:

$$P[j,k] = \begin{cases} \sum_{i=1}^{j-1} d_i c_i + \sum_{i=j}^{k-1} d_i s_i + \sum_{i=k}^{n-1} d_i c_i & \text{if } j\text{th point is red.} \\ \sum_{i=1}^{j-1} d_i c_i + \sum_{i=j}^{k-1} d_i t_i + \sum_{i=k}^{n-1} d_i c_i & \text{if } j\text{th point is blue.} \end{cases}$$

We can find the indices of different colored points $1 \le j < k \le n$ that minimize $P[j,k]$ in $O(n^2)$ time by calculating c_i, s_i, t_i, and d_i for all i in linear time; tabulating the partial sums $\sum_{i=1}^{j-1} d_i c_i$ (for all j) and $\sum_{i=k}^{n-1} d_i c_i$ (for all k) in linear time; and tabulating $\sum_{i=j}^{k-1} d_i s_i$ and $\sum_{i=j}^{k-1} d_i t_i$ (for all pairs $1 \le j < k \le n$) in quadratic time. Once we know the endpoints of the shortest alternating path, we can find the actual path using Lemma 7.

Theorem 8. *Given a set R of n red points and a set B of n blue points such that all points are distinct and collinear, there exists an $O(n^2)$-time algorithm that computes a planar alternating path of minimum length such that no two edges cross and each edge has at most two bends.*

5. Shortest alternating cycles on collinear 3-colored point sets

We extend the results on 2-colored collinear point sets to three colors. We use the colors red, green and blue, denoted by r, g and b. An alternating cycle is a cycle that connects points in the order rgbrgbr etc. We consider the cycle to be oriented in this direction to avoid ambiguity. The following lemma is the 3-color version of Lemma 1.

Lemma 9. *Let R, G and B be sets of n red, n green and n blue points such that all points are distinct and on the x-axis. Let ℓ be a vertical line that intersects the x-axis between two colored points. If there are r red, g green and b blue points to the left of ℓ, then any alternating cycle on $R \cup G \cup B$, crosses ℓ at least $2\max\{1, |r-g|, |g-b|, |b-r|\}$ times.*

Proof. In each component to the left of ℓ, the number of points of one color can be at most one more than the number of points of another color. In addition, since ℓ intersects the x-axis between two colored points, the number of components to the left of ℓ is also at least one. Thus, the minimum number of components of an alternating cycle C to the left of ℓ is $\max\{1, |r-g|, |g-b|, |b-r|\}$. The number of edges of C that intersect ℓ is twice the number of components of C to the left of ℓ. □

We show that given a set of colored points on a horizontal line, we can find an alternating cycle that matches the lower bound of Lemma 9. The proof is by construction and uses a sweep-line algorithm similar to the two color case. However unlike the two color case, the algorithm initially constructs a non-planar embedding of the cycle and only removes crossings at the end.

Assume the red, green and blue points lie on the x-axis, called the *spine*. Let ℓ be a vertical line through a point on the spine that lies between two colored points. Line ℓ sweeps from left to right, and points to the left of ℓ are connected into directed components (directed from red to green to blue to red) that will eventually form an alternating cycle. The components have terminals on the sweep-line. Because a component is directed, the first and last colored points in a component are defined. We say that a component is: a *k-component* with color a if the first and last point of the component have color a; a *kl-component* with color ab if the first two and the last two points of the component have colors a and b; and a *klm-component* with color abc if the first three and the last three points of the component have colors a, b, and c in the same order. For example, Fig. 7(a) shows a kl-component with color br at ℓ_0, two k-components with color r at ℓ_1 and one klm-component with color gbr at ℓ_3.

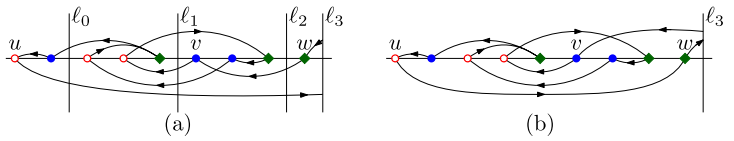

Fig. 7. Left: Several sweep-line positions during the sweep-line algorithm for a 3-colored cycle. Right: The equal length klm-component at ℓ_3 with color brg.

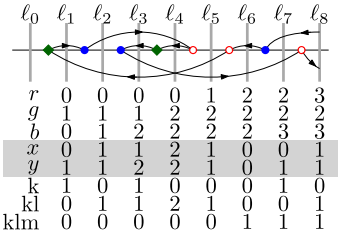

Fig. 8. Illustration of the sweep-line algorithm and the parameter values at the first nine sweep-line positions in a partial sample input. The rows labeled k, kl, and klm contain the number of components of that type.

Let (f_1, f_2, f_3) be the number of occurrences of the least, second most and most frequently occurring color to the left of ℓ, respectively. Let $y = f_3 - f_1$ and $x = f_2 - f_1$, so $0 \le x \le y$. For example in Fig. 7(a), we have $(x, y) = (1, 1)$ at ℓ_0 and $(x, y) = (0, 2)$ at ℓ_1. The algorithm maintains the following invariants:

Q1. If $y > 0$ then there are $y - x$ k-components and x kl-components.
Q2. If $y = 0$ then there is one klm-component and one of its endpoints is the first colored point to the left of ℓ.
Actually, there are always two possible klm-components that satisfy this property. Both have the same length but they have different colors. (In case 2, the algorithm chooses one of these two components to permit the next colored point to connect to an endpoint of the klm-component.) For example, Fig. 7(b) shows a klm-component with color brg that has the same length and endpoint w as the klm-component with color gbr in Fig. 7(a). In general, if u and w are the two colored points adjacent to edges cut by ℓ, with w closest to ℓ and v adjacent to w, we add the edge $\{w, u\}$ (directed appropriately) and remove the edge $\{v, w\}$. This changes the colors of the first three and last three points: rgb $\leftrightarrow$ gbr $\leftrightarrow$ brg $\leftrightarrow$ rgb, following $\rightarrow$ or $\leftarrow$ if (u, w) or (w, u), respectively, is the appropriate direction. Notice that modifying a klm-component twice results in the original component, so only two possible colorings can be obtained. It is important that this modification does not change the length of the component. We remove one edge $\{v, w\}$ and the two partial edges between u and ℓ, and between w and ℓ. We add one edge $\{w, u\}$ and the two partial edges between v and ℓ, and between w and ℓ. Let $d(x, y)$ be the distance between x and y. Since the points are collinear and no colored point (such as u or v) lies between w and ℓ, $d(u, w) + d(w, \ell) = d(u, \ell)$ and $d(v, w) + d(w, \ell) = d(v, \ell)$, so the total length of the component is unchanged.

Lemma 10. *Let R, G and B be sets of n red, n green and n blue points such that all points are distinct and collinear. There exists a $\Theta(n \log n)$-time algorithm that computes an alternating cycle that matches the lower bound of Lemma 9.*

Proof. At sweep-line ℓ, we execute the following algorithm. Assume for ease of presentation that b is the most frequently occurring color and r is the least frequently occurring color. The invariant implies that there are x kl-components with color gb, and there are $y - x$ k-components with color b. Let q be the first colored point to the right of the sweep-line.

1. If q is the leftmost colored point, we form a new k-component consisting of the single point q. Since y increases by one, the invariants are preserved. This case occurs at ℓ_0 to ℓ_1 in Fig. 8.
2. If $y = 0$, then we can pick the klm-component (of the two possible klm-components with equal length) that has one endpoint whose color is the same as q and connect q to its other endpoint. The new component is a k-component, y increases by one, and x doesn't change, so the invariants are preserved. See ℓ_6 to ℓ_7 in Fig. 8.

For the remainder of the cases we assume that $y > 0$.

3. If q has color b, we form a new k-component consisting of the single point q. This adds a k-component, y increases by one, and x doesn't change, so the invariants are preserved. See ℓ_2 to ℓ_3 in Fig. 8.
4. If q has color g and if $x < y$, we connect q to the first point of a k-component, which has color b. This removes a k-component and adds a kl-component with color gb. Since x increases by one, the invariants are preserved. See ℓ_3 to ℓ_4 in Fig. 8.

Fig. 9. Two 4-colored cycles on the same set of eight points.

5. If q has color g and if $x = y$, we form a new k-component consisting of the single point q. This adds one k-component with color g. The color g becomes the most frequent color and the invariants are preserved. Not shown in Fig. 8.
6. If q has color r and if $0 = x < y$, we connect q to the endpoint of a k-component. This removes one k-component and adds one kl-component with color br. The color r becomes the second most frequent color and the invariants are preserved. See ℓ_7 to ℓ_8 in Fig. 8.
7. If q has color r and if $0 < x < y$, we connect q to the last point of a k-component and to the first point of a kl-component. This removes one kl-component (the new component is a k-component). Since both x and y decrease by one, the invariants are preserved. Not shown in Fig. 8.
8. If q has color r and if $1 < x = y$, we connect q to the last point of a kl-component and to the first point of another kl-component. This removes one kl-component (the new component is a kl-component). Since both x and y decrease by one, the invariants are preserved. See ℓ_4 to ℓ_5 in Fig. 8.
9. If q has color r and if $1 = x = y$, we connect q to the last point or to the first point of the single kl-component, thus creating a klm-component with an endpoint at q. (If q is the rightmost point, we connect q to both endpoints and stop.) Notice that the two possible klm-components we could create are equivalent under our modification operation, so it doesn't matter which we choose at this point. (We will modify it to the correct choice once we see the next colored point in case 2.) Since both x and y decrease by one, the invariants are preserved. See ℓ_5 to ℓ_6 in Fig. 8.

The algorithm ensures that the number of cycle components to the left of sweep-line ℓ is one if $y = 0$ (one klm-component) and is y if $y > 0$ ($y - x$ k-components and x kl-components). Since $\max\{1, y\} = \max\{1, |r - g|, |g - b|, |b - r|\}$ for each vertical line ℓ between two colored points (the notation is from Lemma 9), we derive that the cycle produced by the algorithm matches the lower bound at each position of the sweep-line. Thus the length of the alternating cycle produced by the algorithm is optimal.

The time to compute the cycle is dominated by the time to sort the colored points since each step of the sweep-line algorithm takes constant time. □

It is worth remarking that while two edges of the shortest cycle computed by Lemma 10 may cross, a shortest crossing-free alternating cycle can be computed in $O(n^2)$ time and with $O(n)$ bends per edge by using the following result, which is a slight variant of Lemma 1 in [5].

Lemma 11. *Let C be a cycle with n vertices on a collinear point set such that every edge of C has one bend and the edges may cross one another. There exists an $O(n^2)$-time algorithm that removes all edge-crossings from C and introduces at most $O(n)$ bends per edge.*

Proof. Assume the vertices are on the x-axis. Create a path P from the cycle C by removing one edge adjacent to the first (leftmost) vertex. Construct the crossing-free drawing of P using the technique of Chan et al. [5] starting with the first vertex. This technique is based on the following idea: Edges are re-drawn in order along the path; edge (v_i, v_{i+1}) is drawn as a curve from point p_i to point p_{i+1} that stays below all edges drawn so far and above all points p_j with $j > i + 1$.

At the end of this technique, the last vertex in the path has no edge blocking it from below (i.e., it can see the point $y = -\infty$). The first vertex also has no edge blocking it from below, so the last edge of the cycle can be added to the drawing without intersecting any existing edges as a poly-line with one bend.

Since we spend $O(n)$ time to process each edge, the overall time complexity is $O(n^2)$. □

Combining Lemmas 10 and 11, we obtain the following.

Theorem 12. *Let R, G and B be sets of n red, n green and n blue points such that all points are distinct and collinear. There exists an $O(n^2)$-time algorithm that computes a planar alternating cycle of minimum length with $O(n)$ bends per edge.*

It is natural to ask whether one can construct alternating shortest paths and cycles for collinear point sets having more than three colors with an approach that computes drawings which satisfy generalizations of Lemmas 1 and 9. Even for 4-colored point sets such an approach is not possible.

Assume that we have eight points and that we want to embed an alternating cycle with four colors. Fig. 9 shows that for any sweep-line there is a solution that crosses this sweep-line exactly twice. So all lower-bounds are 2. However it is not hard to see that there is no embedding that satisfies all lower-bounds simultaneously.

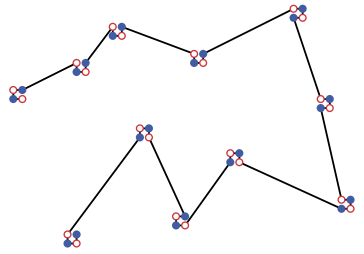

Fig. 10. A small example of the NP-hardness reduction.

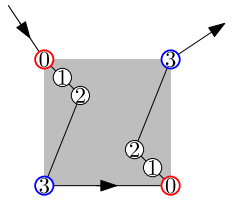

Fig. 11. One square, $S(b)$, in the construction of the point set when the number of colors is $k = 4$. Colors are numbers from 0 to 3.

6. Non-collinear point sets

If the points are non-collinear, we show that finding the shortest alternating cycle is NP-hard by showing that deciding if there is a shortest alternating cycle of length less than L is NP-hard. Our reduction is from the EXACT COVER problem, which is known to be NP-complete [18]: Given a family F of subsets of a finite set U, is there a subfamily F' of F, consisting of disjoint sets, such that $\bigcup_{S \in F'} S = U$?

Theorem 13. *Given a k-colored point set for constant $k \geq 1$, it is NP-hard to find the shortest planar alternating cycle.*

Proof. If $k = 1$, shortest planar alternating cycle is Euclidean TSP, which is NP-hard [20]. If $k = 2$, we describe a polynomial time reduction from EXACT COVER that is a slight modification of the reduction by Papadimitriou [20] showing that Euclidean TSP is NP-hard. Let P be the point set obtained from Papadimitriou's reduction from the EXACT COVER instance, rotated slightly so that no points share the same x- or y-coordinate. If the EXACT COVER instance is solvable, the shortest tour of P has length L (see [20]), while if it is not solvable, the shortest tour of P has length at least $L + \sqrt{a^2+1} - a$ where $a = 20$. Choose positive ϵ to be smaller than the minimum of $(\sqrt{a^2+1} - a)/(10(n+1))$ and half the smallest difference between the x- or y-coordinates of points in P. Let $R = (P + (-\epsilon, -\epsilon)) \cup (P + (\epsilon, \epsilon))$ and $B = (P + (-\epsilon, \epsilon)) \cup (P + (\epsilon, -\epsilon))$ (where $P + (x, y) = \{(p_x + x, p_y + y) | p \in P\}$). That is, each point $p \in P$ becomes a cluster of four points (two red and two blue) forming the corners of a square $S(p)$ of side-length 2ϵ centered at p. See Fig. 10.

If the EXACT COVER instance is solvable, there is a planar alternating tour of $R \cup B$ of length at most $L + 10\epsilon n < L + \sqrt{a^2+1} - a$. The alternating tour follows the shortest tour of P from cluster to cluster. Within the cluster for p, it follows three of the four sides of $S(p)$, leaving one side whose endpoints connect to the two neighbors of the cluster. It is not hard to verify that one may choose such a side for each cluster so that the resulting alternating tour is planar. Indeed, suppose (a, b) and (b, c) are edges of the (cyclically directed) shortest tour of P. Let $s_a \neq s_c$ be two different sides of $S(b)$ such that the interiors of triangles $a \times s_a$ and $c \times s_c$ do not intersect and do not intersect $S(b)$. Such sides must exist since 2ϵ is smaller than the minimum difference between x- or y-coordinates of a, b, and c. Redraw the edge from a to the blue point of s_a and redraw the edge to c from the red point of s_c. Since the triangles do not intersect, the redrawn edges do not intersect. Connect the new endpoints of the redrawn edges by adding the three sides of $S(b)$ that are not adjacent to both new endpoints to the alternating cycle. By repeating this process for all points $b \in P$, we obtain a planar alternating cycle.

The length of this alternating tour is at most $L + (6 + \sqrt{2})\epsilon n < L + 10\epsilon n < L + \sqrt{a^2+1} - a$. If the instance is not solvable, any alternating tour of $R \cup B$ is at least as long as the shortest tour of $R = P$, which has length at least $L + \sqrt{a^2+1} - a$. Thus $R \cup B$ has an alternating tour of length at most $L + \sqrt{a^2+1} - a$ if and only if the EXACT COVER instance is solvable.

If $k > 2$, the reduction is the same except that inside each square are $2(k-2)$ points (two of each color other than red and blue). These points lie on the diagonal that connects the red corners of the square, with one point of color i at distance $i\epsilon/(k-2)$ from each corner, for $i = 1, 2, \ldots, k-2$. (Red is color 0 and blue is color $k-1$.) See Fig. 11.

After following the construction for $k = 2$, for each square $S(b)$, replace each of the two red-to-blue edges on its boundary (keeping the one blue-to-red edge) with $k-2$ edges that follow the diagonal from the red point to its closest point of color 1, then to its closest point of color 2, ... and finally connect the point of color $k-2$ to the blue point. This increases the length of the alternating cycle by at most twice the length of the diagonal minus twice the side length (or $4(\sqrt{2}-1)\epsilon$) per square.

The resulting alternating tour has length at most $L + (5\sqrt{2}+2)\epsilon n$. This is still at most $L + 10\epsilon n < L + \sqrt{a^2+1} - a$, so the reduction still holds. □

7. Open problems

The research in this paper suggests several open problems. We conclude the paper by listing some of those that in our opinion are among the most interesting.

1. Can the time complexity of Theorem 8 be improved?
2. Can the bend-complexity of Theorem 12 be improved? Can that result be extended to paths?
3. The problem of computing shortest alternating cycles on collinear k-colored point sets is open for $k > 3$.
4. Study the problem of drawing not necessarily alternating shortest bicolored cycles/paths on collinear bicolored point sets. That is, we are given a cycle/path where any blue (red) vertex may have a neighbor of its same color and we want to draw the cycle/path using the points of $R \cup B$ such that the total edge length is minimized.
5. Our result assumes that the edges can be drawn "as flat as possible" around the spine. It would be interesting to consider the problem of computing alternating cycles/paths when a minimum distance between any two edges or between any edge and any non-incident vertex must be respected.

References

[1] M. Abellanas, J. Garcia-Lopez, G. Hernández-Peñalver, M. Noy, P.A. Ramos, Bipartite embeddings of trees in the plane, Discrete Appl. Math. 93 (2–3) (1999) 141–148.
[2] J. Akiyama, J. Urrutia, Simple alternating path problem, Discrete Math. 84 (1990) 101–103.
[3] M. Badent, E. Di Giacomo, G. Liotta, Drawing colored graphs on colored points, Theor. Comput. Sci. 408 (2–3) (2008) 129–142.
[4] O. Bastert, S.P. Fekete, Geometrische Verdrahtungsprobleme, Technical Report 96–247, Universität zu Köln, 1996.
[5] T.M. Chan, H.-F. Hoffmann, S. Kiazyk, A. Lubiw, Minimum length embedding of planar graphs at fixed vertex locations, in: S. Wismath, A. Wolff (Eds.), GD 2013, in: LNCS, vol. 8242, Springer-Verlag, 2013, pp. 376–387.
[6] E. Di Giacomo, W. Didimo, G. Liotta, Spine and radial drawings, in: R. Tamassia (Ed.), Handbook of Graph Drawing and Visualization, CRC Press, 2014, pp. 247–280.
[7] E. Di Giacomo, W. Didimo, G. Liotta, H. Meijer, F. Trotta, S.K. Wismath, k-Colored point-set embeddability of outerplanar graphs, J. Graph Algorithms Appl. 12 (1) (2008) 29–49.
[8] E. Di Giacomo, W. Didimo, G. Liotta, H. Meijer, S.K. Wismath, Point-set embeddings of trees with given partial drawings, Comput. Geom. 42 (6–7) (2009) 664–676.
[9] E. Di Giacomo, G. Liotta, F. Trotta, On embedding a graph on two sets of points, Int. J. Found. Comput. Sci. 17 (5) (2006) 1071–1094.
[10] E. Di Giacomo, G. Liotta, F. Trotta, Drawing colored graphs with constrained vertex positions and few bends per edge, Algorithmica 57 (4) (2010) 796–818.
[11] A. Estrella-Balderrama, J.J. Fowler, S.G. Kobourov, Colored simultaneous geometric embeddings and universal pointsets, in: Proceedings of the 21st Annual Canadian Conference on Computational Geometry, Vancouver, British Columbia, Canada, August 17–19, 2009, 2009, pp. 17–20.
[12] F. Frati, M. Glisse, W.J. Lenhart, G. Liotta, T. Mchedlidze, R.I. Nishat, Point-set embeddability of 2-colored trees, in: W. Didimo, M. Patrignani (Eds.), GD 2012, in: LNCS, vol. 7704, Springer-Verlag, 2012, pp. 291–302.
[13] A. Kaneko, M. Kano, Straight-line embeddings of two rooted trees in the plane, Discrete Comput. Geom. 21 (4) (1999) 603–613.
[14] A. Kaneko, M. Kano, Straight line embeddings of rooted star forests in the plane, Discrete Appl. Math. 101 (2000) 167–175.
[15] A. Kaneko, M. Kano, Discrete geometry on red and blue points in the plane – a survey, in: B. Aronov, S. Basu, J. Pach, M. Sharir (Eds.), Algorithms and Combinatorics, in: Discrete Comput. Geom., vol. 25, Springer, 2003, pp. 551–570.
[16] A. Kaneko, M. Kano, Semi-balanced partitions of two sets of points and embeddings of rooted forests, Int. J. Comput. Geom. Appl. 15 (3) (2005) 229–238.
[17] A. Kaneko, M. Kano, K. Suzuki, Path coverings of two sets of points in the plane, in: J. Pach (Ed.), Towards a Theory of Geometric Graphs, in: Contemp. Math., vol. 342, American Mathematical Society, Providence, 2004, pp. 99–112.
[18] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher, J.D. Bohlinger (Eds.), Complexity of Computer Computations, Plenum Press, New York, 1972, pp. 85–103.
[19] J. Pach, R. Wenger, Embedding planar graphs at fixed vertex locations, Graphs Comb. 17 (2001) 717–728.
[20] C.H. Papadimitriou, The Euclidean traveling salesman problem is NP-complete, Theor. Comput. Sci. 4 (1977) 237–244.
[21] M. van Garderen, G. Liotta, H. Meijer, Universal point sets for 2-coloured trees, Inf. Process. Lett. 112 (8–9) (2012) 346–350.