On Embeddability of Buses in Point Sets*

Till Bruckdorfer[1], Michael Kaufmann[1], Stephen Kobourov[2], and Sergey Pupyrev[2]

[1] Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Germany
[2] Department for Computer Science, University of Arizona, USA

Abstract. Set membership of points in the plane can be visualized by connecting corresponding points via graphical features, like paths, trees, polygons, ellipses. In this paper we study the *bus embeddability problem* (BEP): given a set of colored points we ask whether there exists a planar realization with one horizontal straight-line segment per color, called bus, such that all points with the same color are connected with vertical line segments to their bus. We present an ILP and an FPT-approach for the general problem. For restricted versions of this problem, such as when the relative order of buses is predefined, or when a bus must be placed above all its points, we provide efficient algorithms. We show that another restricted version of the problem can be solved using 2-stack pushall sorting. On the negative side we prove the NP-completeness of a special case of BEP.

1 Introduction

Visualization of sets is an important topic in graph drawing and information visualization and the traditional approach relies on representing overlapping sets via Venn diagrams and Euler diagrams [28]. When more than a handful sets are present, however, such diagrams become difficult to interpret and alternative approaches, such as compact rectangular Euler diagrams are needed [27].

Often the geometric position of the elements of the sets are prescribed as points in the plane. The task is to emphasize the sets where the elements belong to. In visualization approaches for set memberships of items on maps, this is done by connecting points from the same set by corresponding lines (LineSets [2]), tree structures (KelpFusion [23]), and enclosing polygons (BubbleSet [11] or MapSets [13]).

We consider a unified version of the tree-structure approach using a model that has been applied before for drawing orthogonal buses known from VLSI design [21]. Our goal is a membership visualization of points in sets by a tree-structure that consists of a single horizontal segment, called *bus*, to which all the points from the same set are connected by vertical segments, called *connections*; see Fig. 1 for planar and non-planar versions. We assume the sets to be given by single-colored points, such that in the final visualization, called *bus realization*, every point of the same color is connected to exactly one bus associated with this color. The objective is to find a position for each bus, such that crossings of buses with connections are avoided, called *planar* bus realization. We call this the *bus embeddability problem* (BEP). Such a simple visualization scheme makes it very easy to recognize the sets and label them, by placing a label inside each bus (if the bus is drawn thick enough), or directly above/next to the bus.

* An extended version including all missing proofs can be found in [7].

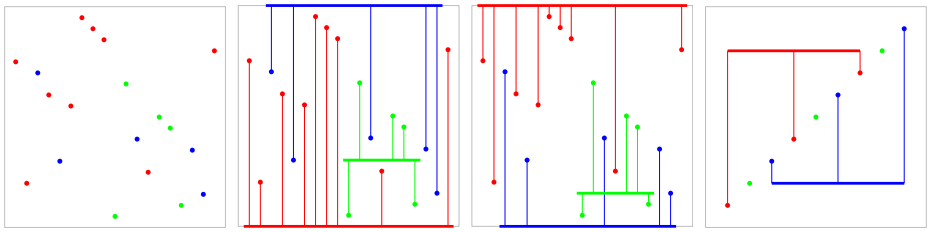

Fig. 1. (a) Fixed positions of points, where points with the same color belong to the same set. (b) A planar bus realization for this setting, while (c) is a non-planar bus realization. (d) A point set without any planar bus realization.

Related Work. Buses have been used, in a more general form, for visualizing degree-restricted hypergraphs. Ada et al. [1] used horizontal and vertical buses in bus realizations, where the points (representing hypervertices contained in at most four hyperedges) were not predefined in the plane. They asked whether a given such hypergraph admits a non-planar bus realizations (allowing connections to cross each other) and showed that the problem is NP-complete. In contrast, if a planar embedding is given, a planar bus realization can be constructed on a $\mathcal{O}(n) \times \mathcal{O}(n)$ grid in $\mathcal{O}(n^{3/2})$ time [6]. These types of problems also have connections to rectangular drawings, rectangular duals and visibility graphs, since the edges of the incidence graph of a hypergraph enforce visibility constraints in the bus realizations [29].

Another related approach is visualization based on graph supports of hypergraphs. Here the goal is to connect the vertices in such a way that each hyperedge induces a connected subgraph [8, 19]. Supported hypergraph visualizations inspired edge-bundling and confluent layouts as alternative visualizations for cliques [14, 12].

A solution to the BEP problem can be viewed as planar tree support for hypergraphs, and this problem is related to Steiner trees, where the goal is to connect a set of points in the plane while minimizing the sum of edge lengths in the resulting tree; this is a classic NP-complete problem [15]. Hurtado et al. [17] considered planar supports for hypergraphs with two hyperedges such that the induced subgraph for every hyperedge and the intersection is a Steiner tree. Their objective was to minimize the sum of edge lengths, while allowing degree one or two for the hypervertices. BEP is even more closely related to rectilinear Steiner trees,where the Euclidean distance is replaced by the rectilinear distance; constructing rectilinear Steiner trees is also NP-complete [16]. A single trunk Steiner tree [10] is a path which contains all vertices of degree greater than one. This is a variant that is solvable in linear time. BEP for a single set is the single trunk rectilinear Steiner tree problem, where we ignore the minimization of the sum of the edge lengths. Thus BEP can be seen as a simultaneous single-trunk rectilinear Steiner tree problem. The fact that a bus placement influences the placement of other buses makes the problem hard.

Consider the input to BEP along with a box that encloses all the points. If in BEP the buses extend to the right boundary of this box, or both to the left and right bound-

ary of this box, then this problem corresponds to backbone boundary labeling [4] and can be efficiently solved. In backbone boundary labeling the problem is to orthogonally connect points by a horizontal backbone segment leading to a label placed at the boundary. In this setting it is always possible to split the problem into two independent subproblems, which is impossible in our case.

BEP is also related to the classical *point set embeddability problem*, where given a set of points along with a planar graph, we need to determine whether there exists a mapping of vertices to points such that the resulting straight-line drawing is planar. The general decision problem is NP-hard [9]. In the variant of orthogeodesic point set embedding, Katz et al. proved that deciding whether a planar graph can be embedded using only orthogonal edge routing is NP-hard [18].

Our Results. In Section 2 we solve BEP when the relative order of the buses is prescribed; we also show that BEP is fixed-parameter tractable with respect to the number of colors. In Section 3 we formulate an ILP that solves BEP and show some experimental results. In Section 4 we restrict BEP (when a bus must be above all its points, or a bus must be either at its topmost or bottommost point) and describe efficient algorithms for these settings. Another restricted version of the problem can be solved using 2-stack pushall sorting. Finally we prove that BEP is NP-complete, even for just two points per color, if points may not lie on buses.

2 Preliminaries

We begin with some definitions. Suppose we are given a set of points $\mathcal{P} = \{p_1, \ldots, p_n\}$ and colors $\mathcal{C} = \{c_1, \ldots, c_k\}$ together with a function $f : \mathcal{P} \longrightarrow \mathcal{C}$, $f(p) = c$. For simplicity, we assume that no two points share a coordinate in the input point set, although in some illustrations the input points might violate this assumption. The bus embeddability problem (BEP) asks, whether there is a planar bus realization with one horizontal bus per color. BEP is a decision problem, but in our descriptions whenever the answer is affirmative we also compute a drawing. We refer to such a drawing as a *solution of BEP*. In the negative case we say that BEP has no solution.

A point p has x-coordinate $x(p)$ and a y-coordinate $y(p)$, as well as a color $f(p)$. In a bus realization we have connections only between a point p and a bus c of the same color, that is, $c = f(p)$. We denote by $f^{-1}(c)$ the set of points with color c. Bus c naturally extends from the x-coordinate $x_l(c) = \min\{x(p) | p \in f^{-1}(c)\}$ of the leftmost point to the x-coordinate $x_r(c) = \max\{x(p) | p \in f^{-1}(c)\}$ of the rightmost point of $f^{-1}(c)$. We call $[x_l(c), x_r(c)]$ the *span* of c, which is predefined by the input points. The y-coordinate of a bus c is denoted by $y(c)$, which is the only parameter to be determined for a solution for BEP.

Note that BEP is trivial when there are at most two colors: it is always possible to place one bus at the top and the other (if exists) at the bottom of the drawing. Thus in the following we assume $k > 2$. For more than two colors, the relative order of the buses is important; see Fig. 1. Suppose the y-order of the buses is prescribed. The next lemma shows that one can check an existence of a solution for BEP respecting the order.

Lemma 1. *There is a $\mathcal{O}(n \log n)$-time algorithm that, given an order of buses, tests whether there exists a solution for BEP respecting the order.*

Proof. Suppose we are given an order $c_1 < \cdots < c_k$ of the buses from bottom to top. We use discrete values for the y-coordinates increasing from bottom to top, where a unit is $1/n$ of the y-distance of two consecutive points. We first present a simpler $\mathcal{O}(n^2)$-time algorithm, and then describe how to speed it up.

Recall that the span of every bus is defined by an input point set; hence, we only show how to choose y-coordinates of the buses. The first bus, c_1, is placed at y-coordinate $y(c_1) = 0$, and all the points of color c_1 are connected to the bus. Assume that bus c_{i-1} is placed at y-coordinate $y(c_{i-1})$ and is connected to all its points. We place c_i at $y(c_i) = y(c_{i-1}) + 1$ unit and check if the bus crosses a previously drawn (vertical) segment. If it does cross a segment, then we shift c_i one unit upwards by increasing $y(c_i)$ and repeat the procedure. Once the bus is placed without crossings, we connect it to the corresponding points. Consider the vertical segment of a point p of color c_i. It is easy to see that if $y(p) \geq y(c_i)$, then the segment cannot cross a previously placed bus c_j for $j < i$. If $y(p) < y(c_i)$ and the vertical segment crosses a bus, then such a crossing is unavoidable in any solution respecting the given order. Hence, we may stop the algorithm reporting that no solution exists. Otherwise, we proceed with the next color.

The above algorithm can easily be implemented in quadratic time. However, we can do better using the following observation: Every bus is placed at its bottommost "valid" y-coordinate, that is, the one that does not produce crossings with previously placed buses. To find such a y-coordinate efficiently for each color, we store all points of the already processed colors in a data structure D that supports the range operation such as "extracting minimum/maximum on a given range". For every color c_i, we extract a point with the maximum y-coordinate in the range corresponding to the span of c_i. The bus of c_i is placed at the maximum of the extracted y-coordinate and the y-coordinate of bus $y(c_{i-1})$. Then all the points of color c_i are added to D. A balanced tree (e.g., a segment tree) providing logarithmic complexity for insert and extract operations is sufficient for our needs. □

In general the correct order of the buses for a planar bus realization is not known. One can apply Lemma 1 for each of the $k!$ possible bus orders, which yields an $\widetilde{\mathcal{O}}(k!)$-time[3] algorithm for BEP. Next, we improve the running time with an algorithm providing deeper insight into the structure of the problem.

[1]: **TB:** I added a reason for Lemma 2

Lemma 2. *There is a $\widetilde{\mathcal{O}}(2^k)$-time algorithm for BEP.*

Proof. We solve a given instance of BEP using dynamic programming. Let us call a *state* a pair (h, B), where $0 \leq h \leq n + 1$ is an integer and B is a subset of $\mathcal{C} = \{c_1, \ldots, c_k\}$. By a solution for a state (h, B) we mean a (planar) bus realization consisting of buses for every color $c \in B$ such that the topmost bus has y-coordinate h. If such a solution exists, we write $F(h, B) = \mathrm{true}$, and otherwise $F(h, B) = \mathrm{false}$. It is easy to see that a solution for the original BEP problem exists if and only if $F(h, \mathcal{C}) = \mathrm{true}$ for some $0 \leq h \leq n + 1$.

[3] $\widetilde{O}$ hides polynomial factors.

We reduce the problem to solving it for "smaller" states, that are the states with fewer elements in B. As a base case, we set $F(h, B) = \text{true}$ for all $0 \leq h \leq n+1$ and $|B| = 1$. To compute a value for a state $F(h, B)$ with $|B| > 1$, we consider a color $c^* \in B$. Let $h^* = \max\{y(p) | f(p) \in B \setminus \{c^*\}$ and $x_l(c^*) \leq x(p) \leq x_r(c^*)\}$, that is, the largest (topmost) y-coordinate of a point of color $B \setminus \{c^*\}$ laying in the span of c^*. It follows from the proof of Lemma 1 that the bus for c^* should be placed at y-coordinate h^*. Thus, $F(h, B)$ is set to true if (a) $h \geq h^*$ and (b) there exists a solution for a state $(h', B \setminus \{c^*\})$ for some $h' < h$. We stress here that in order to compute $F(h, B)$, one needs to consider every color of B as a potential c^*. There are $n2^k$ different states, and a computation for a single state clearly takes a polynomial number of steps. □

The above result shows that the BEP problem is fixed-parameter tractable with respect to k, that is, it can be efficiently solved for a small number of buses. Note that in Section 5 we prove that BEP is NP-complete; hence, it is unlikely that a polynomial-time (in terms of k) algorithm exists.

3 An ILP for BEP

In this section we present an integer linear programming (ILP) formulation for BEP that produces a planar bus realization if one exists. Here we also minimize the amount of ink, which is the sum of all segment lengths.

Lemma 3. *A solution for BEP can be computed by an ILP.*

Proof. In a preprocessing step we compute the span of every bus $c \in \mathcal{C}$. It remains to compute the y-coordinate variable $y(c)$ of every bus c. To this end, we introduce a planarity constraint for every point $p \in \mathcal{P}$ within the span of bus c having a different color. The pairs $(p, c), c \neq f(p)$ are called *conflicting*. Conflicting pairs (p, c) are stored in a matrix $\mathcal{J}$ and induce the constraint $(y(p) < y(c)$ and $y(f(p)) < y(c))$ or $(y(p) > y(c)$ and $y(f(p)) > y(c))$. The matrix $\mathcal{J}$ can be computed in $\mathcal{O}(kn)$ time, where n=$|\mathcal{P}|$ and k=$|\mathcal{C}|$. In order to minimize the amount of ink, we sum up the lengths of all connections and ignore the lengths of buses, as those are determined by the input.

$$\begin{aligned} \min \quad & \sum_{c \in \mathcal{C}} \sum_{f(p)=c} |y(c) - y(p)| \\ s.t. \quad & (y(p) < y(c) \vee y(f(p)) > y(c)) \wedge (y(p) > y(c) \vee y(f(p)) < y(c)) \quad \forall (p, c) \in \mathcal{J} \\ & 0 \leq y(c) \leq \max_{p \in \mathcal{P}} \{y(p)\} + 1 \end{aligned}$$

Since absolute value (resp. "or") needs one more variable and 3 constraints for every point (resp. for every conflicting pair) [4], the final ILP has n+k+2$|\mathcal{J}|$ variables and 3n+k+6$|\mathcal{J}|$ constraints. □

In order to get a feeling about the probability that a point set admits a solution of BEP, we ran a small experiment with the ILP, implemented with the Gurobi solver [24].

[4] $\min \sum |a-b| \Leftrightarrow \min \sum e, e \geq a-b, e \geq b-a, e \geq 0$; $(a < b) \vee (c < d) \Leftrightarrow a - b < eM, c - d < (1-e)M, e \in \{0, 1\}, M = \infty$

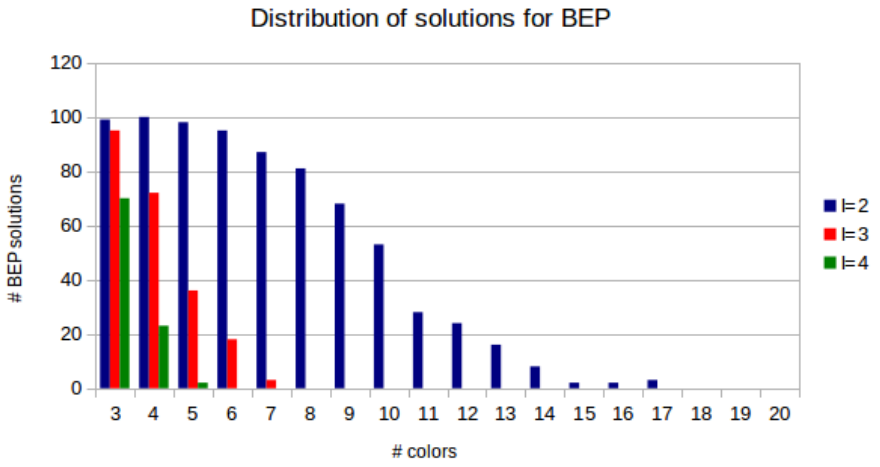

Fig. 2. The percentage of solutions for BEP for a random point set of size $n = kl$ with $l = 2, 3, 4$ points per color out of $k = 3, \ldots, 20$ colors.

We considered point sets with $k = 3, \ldots, 20$ colors and with $l = 2, 3, 4$ points per color. We randomly placed the points on a 1024×768 area. For each pair (l, k) we counted the number of BEP solutions out of 100 instances; see Fig. 2. The remaining instances were infeasible. For a fixed number of points l, the number solutions for BEP decreases with increasing number of colors k. It decreases faster the higher l is. On the other hand for a fixed number of colors k, the number of solutions for BEP also decreases with increasing number of points l. Thus, even studying two points per color promises to be sufficiently interesting. Thus, as base case for further analysis, we initially consider two points per color, before dealing with the general case, where in real instances solutions rarely exist. It is possible that many more solutions exist if we allow only few crossings, but all non-planar settings are left as open problems.

[2]: **TB:** addressed that real instances have rarely a solution

[3]: **TB:** addressed also here (apart from the open problem part) a comment on non-planar settings

4 Efficiently Solvable BEP Variants

In this section we consider three variants of BEP, which can be solved in polynomial time. A bus c is called *top* (resp., *bottom*) if all of its points are below (resp., above) the bus, that is, $y(c) \geq y(p)$ (resp., $y(c) \leq y(p)$) for all $p \in f^{-1}(c)$. We distinguish between buses that are above (below) of their points and buses that pass through one of their points. A top-bus is a $\sqcap$-*bus* if $y(c) > y(p)$ for all $p \in f^{-1}(c)$ (Fig. 3(a)), while it is a $\urcorner$-*bus* if $y(c) = y(p)$ for a point p with $y(p) = \max\{y(q) | q \in f^{-1}(c)\}$ (Fig. 3(c)). Similarly we define a $\sqcup$-*bus* and a $\llcorner$-*bus*; see Figs. 3(b) and 3(d). A bus, whose type is none of the four types from above, is called a *center-bus*. The variant of BEP where only buses of the types in $S \subseteq \{\sqcap, \sqcup, \urcorner, \llcorner\}$ are allowed to use is denoted by S-BEP.

[4]: **TB:** I added the definition of general variants for BEP

In Section 4.1 we study $\sqcap$-buses and provide an algorithm for $\sqcap$-BEP. The same algorithm obviously solves the $\sqcup$-BEP variant. Next we consider $\urcorner$-buses and $\llcorner$-buses.

Note that $\sqcap$-BEP and $\sqcup$-BEP are trivial, since every $\sqcap$-bus (resp., $\sqcup$-bus) is uniquely defined by its span and the topmost (bottommost) point. Hence, we investigate and design an efficient algorithm for the $(\sqcap, \sqcup)$-BEP variant. Finally in Section 4.3, we examine the general BEP for a specific point set, where all points lie on a diagonal. We show that the variant of the problem is equivalent to a longstanding open problem (resolved very recently) of sorting a permutation with a series of two stacks.

4.1 $\sqcap$-BEP

We present an algorithm that decides in polynomial time whether a drawing with $\sqcap$-buses exists for a given input, and constructs such a drawing if one exists.

Theorem 1. *There exists an $\mathcal{O}(n \log n)$-time algorithm for $\sqcap$-BEP.*

Proof. For ease of presentation, we first assume that the input consists of two points per color, that is, $k = n/2$, and provide a simple quadratic-time implementation. Later we generalize the algorithm and improve the running time. Intuitively, the algorithm sweeps a line from bottom to top and processes the points in increasing order of y-coordinates. At every step, we keep all the vertical segments of the "active" colors (the ones without a bus) in the correct left-to-right order. If two vertical segments of the same color are adjacent in the order, then we can draw the corresponding bus and remove the color and its vertical segments. Otherwise, all the active vertical segments have to be "grown" until we reach the next point. It is easy to see that a solution exists if and only if the set of active colors is empty after processing all the points.

More formally, the points are processed one-by-one in increasing order of their y-coordinates. The points are stored in an array sorted by x-coordinate, that is, we have $(p_1, \ldots, p_n)$ with $x(p_1) < \cdots < x(p_n)$. At each iteration, a new point is inserted into the array in the position determined by its x-coordinate. Then the array is modified (or simplified) so that the pairs of points of the same color that are adjacent in the array are removed. That is, if $f(p_i) = f(p_{i+1})$ for some $1 \leq i < n$, then we get a new array $(p_1, \ldots, p_{i-1}, p_{i+2}, \ldots, p_n)$. The simplification is performed as long as the array contains monochromatic adjacent points. After this step the algorithm proceeds with the next point. For every color c, we keep the value $y^*(c)$, which is equal to the y-coordinate $y(p), p \in f^{-1}(c')$ of the point of color c', whose insertion into the array induced the removal of points $f^{-1}(c)$ from the array. If the algorithm ends up with a non-empty array, then we report that no solution exists. Otherwise, the y-coordinate of the resulting bus of color c is $y^*(c) + \varepsilon$, where $\varepsilon > 0$ is sufficiently small to avoid overlaps between the buses. An example of the algorithm is illustrated in Fig. 4.

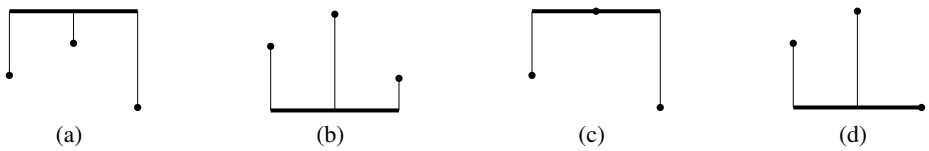

Fig. 3. Illustration of (a) $\sqcap$-bus, (b) $\sqcup$-bus, (c) $\sqcap$-bus, and (d) $\sqcup$-bus.

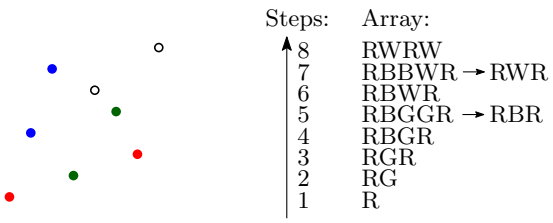

Fig. 4. Running the algorithm from Lemma 1 on a given point set with red (R), green (G), blue (B), and white (W) pairs of points. Since the resulting array is not empty, there is no solution for the instance. Notice that removing any of the colors yields an instance with a solution.

Correctness. The correctness follows from the observation that the algorithm chooses the lowest "available" y-coordinate for every bus, that is, the one that does not induce a crossing between the bus and vertical segments of other colors. Indeed, if at any step of the algorithm we get a color pattern $R, \ldots, B, \ldots, R$ in the array formed by red (R) and blue (B) points and the second blue point p has not been processed yet, then clearly in any solution the red vertical segments reach the y-coordinate of p. Hence, it is safe to "grow" the segments. On the other hand, if processed points form a color pattern RR (that is, two consecutive points of the same color), then there is a solution connecting the corresponding vertical segments at the current y-coordinate. The two points can be removed from consideration, as they cannot create crossings with the subsequent buses. It is also easy to see that the algorithm minimizes ink of the resulting drawing.
Running time. At every iteration of the algorithm, we need to insert a new point into the sorted array and then run the simplification procedure. Point insertion takes $\mathcal{O}(n)$ time and removal of a pair of points from the array can also be done in $\mathcal{O}(n)$ time. Since every pair is removed only once, the total running time is $\mathcal{O}(n^2)$.

To get down to $\mathcal{O}(n \log n)$ time, we use a balanced binary tree instead of an array to store the points. The tree is sorted by the x-coordinates of the points; hence, insertion/removal of a point takes $\mathcal{O}(\log n)$ time. Note that after inserting/removing a point, the only potential candidate pairs for simplification are the point's neighbors that can be found in $\mathcal{O}(\log n)$ time. Again, every point is inserted/removed only once; thus, the total running time is $\mathcal{O}(n \log n)$.

Finally, we observe that the algorithm can be generalized to handle multiple points per color. To this end, we change the simplification step so that the points are removed only if they form a contiguous subsequence in the array (tree), containing all points of this color. Hence we need to know the number of points for each color, which can be done with a linear-time scan of the input. It is easy to see that the proof of correctness can be appropriately modified and the running time remains the same. □

4.2 $(\sqsubset, \sqsubset)$-BEP

We present an algorithm that decides in polynomial time whether $(\sqsubset, \sqsubset)$-BEP has a solution for a given input, and constructs a drawing if one exists.

Theorem 2. *There exists an $\mathcal{O}(n^2)$-time algorithm for $(\sqsubset, \sqsubset)$-BEP.*

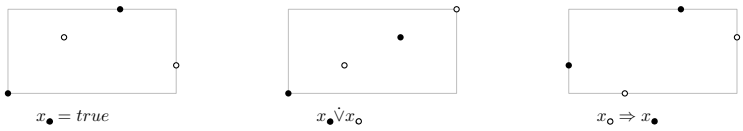

Fig. 5. Three examples for creating clauses for two colors black and white.

Proof. The span of every bus is predefined by the input, while the y-coordinate has precisely two options. We show that $(\ulcorner, \llcorner)$-BEP can be modeled by 2-SAT, and thus is efficiently solvable. For ease of presentation, we assume that the input consists of two points per color and describe a simple quadratic-time algorithm.

The algorithm creates a variable x_c for every color $c \in \mathcal{C}$. The value of x_c is *true* if c is a $\ulcorner$-bus, and it is *false* if c is a $\llcorner$-bus. After that for every pair of colors c, c', the algorithm creates a clause for the 2-SAT instance when the corresponding buses induce a crossing. Building the clauses with respect to the relative position of points is a straight-forward procedure; 3 examples are illustrated in Fig. 5. We can generalize this idea in a straight-forward manner to the case of more points per color. In the general case the y-coordinate of a bus still has precisely two options. In contrast to the case with two points per color we check several points (not only the leftmost or rightmost point) of color c' for their position with respect to the points of color c, since points lie not necessarily in corners of the enclosing rectangle.[5]

[5]: **TB:** I added a sentence for many points per color

Correctness. The correctness follows from the complete case analysis.

Running time. We remark that for the $n^2/4$ pairs of colors, we create $\mathcal{O}(n^2)$ clauses, each clause in constant time by a case analysis. This results in a 2-SAT instance with k variables $x_c, c \in \mathcal{C}$ and $\mathcal{O}(n^2)$ clauses. We solve this instance in linear time [3] and the solution determines the drawing: c is drawn as a $\ulcorner$-bus, if the value of x_c is *true*, otherwise c is drawn as a $\llcorner$-bus. □

4.3 Diagonal BEP

Here we consider a *diagonal* point set in which all points lie on a single diagonal line and there are two points per color. We assume that the point set is *separable*, that is, there is a straight line separating every pair of points having the same color; see Fig. 6. This specific arrangement can be naturally described in terms of permutations. Assuming that the colors are numbered from 1 to k in the order along the diagonal from bottom to top, the input is described by a permutation $\pi = [\pi(1), \ldots, \pi(k)]$ on $\{1, \ldots, k\}$. Such an instance is called *diagonal* π-BEP.

It turns out that this variant of BEP is closely related to the well-studied topic of sorting a permutation with stacks introduced by Knuth in the 1960's [20]. We next show that diagonal π-BEP has a solution if and only if π can be sorted with 2 stacks in series. The problem of deciding whether a permutation is sortable with 2 stacks in series is a longstanding open problem and it has been conjectured to be NP-complete several times [5]. Only very recently a polynomial-time algorithm has been developed [26, 25]. It is an indication that even our restricted variant of BEP is highly non-trivial. Next we sketch a proof of the equivalence.

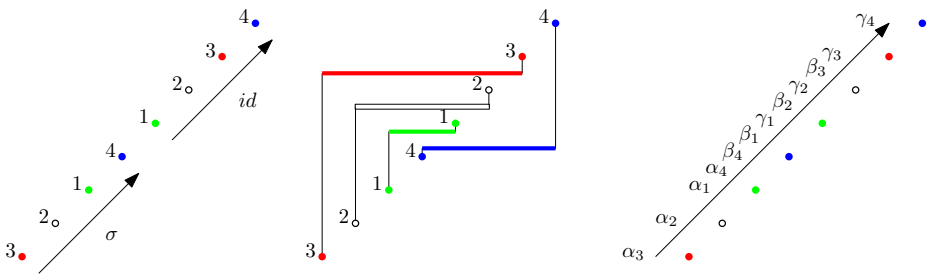

Fig. 6. A diagonal point set with a solution for BEP and the regarding sorting sequence.

First observe that for a diagonal point set with 2 points per color, a top-bus (bottom-bus) can be transformed to a center-bus. For every color c, there are no points of different color within the span of c above the topmost point of c. Hence, we may only consider center-buses in the variant of BEP. For the 2-stack sorting problem, given a permutation π, we want to sort the numbers to the identity permutation $[1, \ldots, k]$ with two stacks S_I, S_{II} using the following operations:

- α_i : read the next element i from input π and push it on the first stack S_I;
- β_i : pop the topmost element i from S_I and push it on S_{II};
- γ_i : pop the topmost element i from S_{II} and print it to the output.

To make the equivalence between 2-stack sorting and bus embeddability, we note that the first operation, α_i, corresponds to the left vertical segment of color i, the second one, β_i, is the bus of i, while γ_i corresponds to the right vertical segment of the color; see Fig. 6. A crossing in the drawing correspond to an "invalid" sorting operation in which either a non-topmost element is moved from S_I to S_{II} (a crossing to the "left" of the diagonal), or a non-topmost element is moved from S_{II} to the output (a crossing to the "right" of the diagonal). Hence, sorting sequences of the operations for π are in one-to-one correspondence with planar bus realization for the point set. Since the point set is separable, all the elements of π will be pushed to S_I before any of the elements is popped to the output. This is called 2-stack *pushall* sorting, see [25] for more details.

Theorem 3. *Diagonal π-BEP has a solution if and only if π is 2-stack pushall sortable. This can be checked in $\mathcal{O}(n^2)$ time.*

5 Hardness of BEP

We sketch the idea behind the proof that BEP$^{\varepsilon}$ for 2 points per color is NP-complete, where BEP$^{\varepsilon}$ is BEP with minimum distance ε of points to their bus as additional input.

We can easily verify a possible solution using Lemma 1; thus BEP$^{\varepsilon}$ is in the class NP. To prove the hardness of BEP$^{\varepsilon}$, we reduce from planar 3-SAT [22], which is 3-SAT, where an instance is represented by a graph whose vertices represent variables and clauses and whose edges represent containment of variables in clauses. We replace the vertices and edges by gadgets. We first restrict ourselves to $(\sqcap, \sqcup)$-BEP and drop the "no points share a coordinate" restriction. Details can be found in [7].

The most important module of the construction is a *chain link*, which is also a gadget for replacing variables. It consists of two points on a common horizontal line that will be connected by a bus. We replace the edges of the graph by chains consisting of nested chain links and replace the clause vertices by a big construction of points, that allows two specific points to be connected via a bus using only one of three choices. We use the input ε to be able to block some choices for this bus. We finally transform the construction into the "no points share a coordinate" setting and allow also center-buses.

Theorem 4. *BEP$^{\varepsilon}$ for 2 points per color is* NP*-complete.*

6 Conclusion and Future Work

We studied bus embeddability, where a set of colored points is covered by a set of horizontal buses, one per color and without crossings. We described an ILP and an FPT-algorithm for the general problem and presented efficient algorithms for several restricted versions. The general problem is shown to be NP-complete even for two points per color when points may not lie on buses.

It is still open to determine the complexity of BEP in the following cases:

- BEP using only center-buses;
- $(\sqcap, \sqcup)$-BEP, that is, BEP without center-buses;
- the general diagonal BEP, with more than 2 points per color;
- the general BEP (since we used an extra ϵ as a parameter).

A natural generalization would be to allow both horizontal and vertical buses, as in [1, 6]. Another variant might be to consider multi-colored points, where a point has to be connected either to all the buses of its corresponding colors, or to at least one of them. For point sets that have no solution for BEP with only one bus per color, we may allow more than one bus or bound the number of crossings. Possible objectives in these scenarios are to minimize the total number of buses over all colors, to minimize the total number of buses, or to minimize the total number of buses if each tree can connect $\leq$ k unicolored points. These objectives are even interesting if a solution to BEP exists.

[6]: **TB:** I addressed the comment on crossings

[7]: **TB:** I didn't address the comment of the variant, whether always a solution exists if we bound the number of crossings. But we know that if the bound is small we can construct a counterexample with n crossings.

[8]: **TB:** We deleted 5 references: He, Hwang, Thompson, Brandes, Ganley and added a reference to an ArXiv version

References

1. Ada, A., Coggan, M., Marco, P.D., Doyon, A., Flookes, L., Heilala, S., Kim, E., Wing, J.L.O., Préville-Ratelle, L.F., Whitesides, S., Yu, N.: On bus graph realizability. CCCG abs/cs/0609127, 229–232 (2007)
2. Alper, B., Riche, N.H., Ramos, G., Czerwinski, M.: Design study of LineSets, a novel set visualization technique. IEEE Trans. Vis. Comput. Graph. 17(12), 2259–2267 (2011)
3. Aspvall, B., Plass, M.F., Tarjan, R.E.: A linear-time algorithm for testing the truth of certain quantified Boolean formulas. Information Processing Letters 8(3), 121–123 (1979)
4. Bekos, M.A., Cornelsen, S., Fink, M., Hong, S.H., Kaufmann, M., Nöllenburg, M., Rutter, I., Symvonis, A.: Many-to-one boundary labeling with backbones. In: Wismath, S., Wolff, A. (eds.) Graph Drawing. LNCS, vol. 8242, pp. 244–255. Springer (2013)
5. Bóna, M.: A survey of stack-sorting disciplines. Electronic Journal of Combinatorics (2) (2002)

6. Bruckdorfer, T., Felsner, S., Kaufmann, M.: On the characterization of plane bus graphs. In: Spirakis, P.G., Serna, M.J. (eds.) CIAC. LNCS, vol. 7878, pp. 73–84. Springer (2013)
7. Bruckdorfer, T., Kaufmann, M., Kobourov, S., Pupyrev, S.: On embeddability of buses in point sets. CoRR - (2015)
8. Buchin, K., van Kreveld, M.J., Meijer, H., Speckmann, B., Verbeek, K.: On planar supports for hypergraphs. J. Graph Algorithms Appl. 15(4), 533–549 (2011)
9. Cabello, S.: Planar embeddability of the vertices of a graph using a fixed point set is NP-hard. J. Graph Alg. Appl. 10(2), 353–366 (2006)
10. Chen, H., Qiao, C., Zhou, F., Cheng, C.K.: Refined single trunk tree: A rectilinear Steiner tree generator for interconnect prediction. In: SLIP. pp. 85–89. ACM (2002)
11. Collins, C., Penn, G., Carpendale, M.S.T.: Bubble Sets: Revealing set relations with iso-contours over existing visualizations. IEEE Trans. Vis. Comput. Graph. 15(6), 1009–1016 (2009)
12. Dickerson, M., Eppstein, D., Goodrich, M.T., Meng, J.Y.: Confluent drawings: Visualizing non-planar diagrams in a planar way. J. Graph Algorithms Appl. 9(1), 31–52 (2005)
13. Efrat, A., Hu, Y., Kobourov, S.G., Pupyrev, S.: MapSets: visualizing embedded and clustered graphs. In: Graph Drawing. pp. 452–463. Springer (2014)
14. Gansner, E.R., Koren, Y.: Improved circular layouts. In: Kaufmann, M., Wagner, D. (eds.) Graph Drawing. LNCS, vol. 4372, pp. 386–398. Springer (2006)
15. Garey, M.R., Graham, R.L., Johnson, D.S.: The complexity of computing Steiner minimal trees. SIAM Journal on Applied Mathematics 32(4), 835–859 (1977)
16. Garey, M.R., Johnson, D.S.: The rectilinear Steiner tree problem is NP-complete. SIAM Journal on Applied Mathematics 32(4), 826–834 (1977)
17. Hurtado, F., Korman, M., van Kreveld, M.J., Löffler, M., Adinolfi, V.S., Silveira, R.I., Speckmann, B.: Colored spanning graphs for set visualization. In: Wismath, S.K., Wolff, A. (eds.) Graph Drawing. LNCS, vol. 8242, pp. 280–291. Springer (2013)
18. Katz, B., Krug, M., Rutter, I., Wolff, A.: Manhattan-geodesic embedding of planar graphs. In: Graph Drawing, 17th International Symposium, GD 2009, Chicago, IL, USA, September 22-25, 2009. Revised Papers. pp. 207–218 (2009)
19. Klemz, B., Mchedlidze, T., Nöllenburg, M.: Minimum tree supports for hypergraphs and low-concurrency Euler diagrams. In: SWAT. LNCS, vol. 8503, pp. 265–276. Springer (2014)
20. Knuth, D.E.: The Art of Computer Programming, Volume 1 (3rd Ed.): Fundamental Algorithms. Addison Wesley Longman Publishing Co., Inc. (1997)
21. Lengauer, T.: VLSI theory. In: Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A), pp. 835–868 (1990)
22. Lichtenstein, D.: Planar formulae and their uses. SIAM Journal on Computing 11(2), 329–343 (1982)
23. Meulemans, W., Riche, N.H., Speckmann, B., Alper, B., Dwyer, T.: KelpFusion: A hybrid set visualization technique. Visualization and Computer Graphics, IEEE Transactions on 19(11), 1846–1858 (2013)
24. Optimization, I.G.: Gurobi optimizer reference manual (2015), `www.gurobi.com`
25. Pierrot, A., Rossin, D.: 2-stack pushall sortable permutations. CoRR abs/1303.4376 (2013)
26. Pierrot, A., Rossin, D.: 2-stack sorting is polynomial. In: Mayr, E.W., Portier, N. (eds.) STACS. LIPIcs, vol. 25, pp. 614–626 (2014)
27. Riche, N.H., Dwyer, T.: Untangling Euler diagrams. Visualization and Computer Graphics, IEEE Transactions on 16(6), 1090–1099 (2010)
28. Simonetto, P., Auber, D., Archambault, D.: Fully automatic visualisation of overlapping sets. Computer Graphics Forum 28(3), 967–974 (2009)
29. Tamassia, R., Tollis, I.G.: A unified approach to visibility representations of planar graphs. Discrete & Computational Geometry 1, 321–341 (1986)