Geometric Simultaneous Embeddings of a Graph and a Matching

Sergio Cabello[1], Marc van Kreveld[2], Giuseppe Liotta[3], Henk Meijer[4], Bettina Speckmann[5], and Kevin Verbeek[5]

[1] Faculty of Mathematics and Physics, University of Ljubljana, Slovenia
sergio.cabello@fmf.uni-lj.si
[2] Dep. of Computer Science, Utrecht University, The Netherlands
marc@cs.uu.nl
[3] Dip. di Ing. Elettronica e dell'Informazione, Università degli Studi di Perugia, Italy
liotta@diei.unipg.it
[4] Roosevelt Academy, Middelburg, The Netherlands
h.meijer@roac.nl
[5] Dep. of Mathematics and Computer Science, TU Eindhoven, The Netherlands
speckman@win.tue.nl, k.a.b.verbeek@tue.nl

Abstract. The geometric simultaneous embedding problem asks whether two planar graphs on the same set of vertices in the plane can be drawn using straight lines, such that each graph is plane. Geometric simultaneous embedding is a current topic in graph drawing and positive and negative results are known for various classes of graphs. So far only connected graphs have been considered. In this paper we present the first results for the setting where one of the graphs is a matching.

In particular, we show that there exists a planar graph and a matching which do not admit a geometric simultaneous embedding. This generalizes the same result for a planar graph and a path. On the positive side, we describe algorithms that compute a geometric simultaneous embedding of a matching and a wheel, outerpath, or tree. Our proof for a matching and a tree sheds new light on a major open question: do a tree and a path always admit a geometric simultaneous embedding? Our drawing algorithms minimize the number of orientations used to draw the edges of the matching. Specifically, when embedding a matching and a tree, we can draw all matching edges horizontally. When embedding a matching and a wheel or an outerpath, we use only two orientations.

1 Introduction

The computation of node-link diagrams of two sets of relations on the same set of data is a recent and already well-established research direction in network visualization. The interest in this problem is partly due to its theoretical relevance and partly motivated by its importance in many application areas, such as software engineering, data bases, and social networks. There are various application scenarios where a visual analysis of dynamic and evolving graphs defined on the same set of vertices is useful, see [3,4] for detailed descriptions.

Formally, the problem can be stated as follows: Let G_1 and G_2 be two graphs that share their vertex set, but which have different sets of edges. We would like to compute two readable drawings of G_1 and G_2 such that the locations of the vertices are the same in both visualizations. Cognitive experiments [9] prove that the readability of a drawing is negatively affected by the number of edge crossings and by the number of bends along the edges. Hence, if G_1 and G_2 are both planar, we want to compute plane drawings of the two graphs where the vertices have the same locations and edges are straight-line segments. Note that we allow edges from different graphs to cross.

In a seminal paper, Brass *et al.* define a *geometric simultaneous embedding* of two planar graphs sharing their vertex set as two crossing-free straight-line drawings that share the locations of their vertices [1]. Geometric simultaneous embedding is a current topic in graph drawing and positive and negative results are known for various classes of graphs. A comprehensive list can be found in Table 1 of a recent paper by Frati, Kaufmann, and Kobourov [7]. Specifically, Brass *et al.* [1] show that two paths, two cycles, and two caterpillars always admit a geometric simultaneous embedding. (A caterpillar is a tree such that the graph obtained by deleting the leaves is a path.) The authors also prove that three paths may not admit a geometric simultaneous embedding. Erten and Kobourov [5] prove that a planar graph and a path may not admit a geometric simultaneous embedding. Frati, Kaufmann, and Kobourov [7] extend this negative result to the case where the path and the planar graph do not share any edges. Geyer, Kaufmann, and Vrt'o [8] show that two trees may not have a geometric simultaneous embedding. A major open question in this area is the following: do a tree and a path always admit a geometric simultaneous embedding? Finally, Estrella-Balderrama *et al.* [6] prove that determining whether two planar graphs admit a geometric simultaneous embedding is NP-hard.

So far, only connected graphs have been considered and in particular, there are no results for one of the simplest classes of graphs, namely *matchings*. A matching is an independent set of edges. Clearly a geometric simultaneous embedding of two matchings always exists, since the union of two matchings is a collection of cycles and hence planar. But already the union of the edges of a path and a matching does not have to be planar: see Fig. 1 (left), which shows a path and a matching which form a subdivision of $K_{3,3}$.

Results. We study geometric simultaneous embeddings of a matching with various standard classes of graphs. In Section 2 we show that there exists a planar

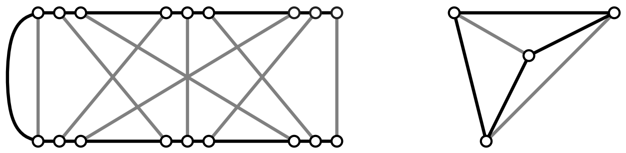

Fig. 1. Left: The union of a path (black) and a matching (gray), can be non-planar. Right: Two orientations of the matching edges (gray) are forced.

graph and a matching which do not admit a geometric simultaneous embedding. This generalizes the same result for a planar graph and a path [5].

On the positive side, we describe algorithms that compute a geometric simultaneous embedding of a matching and a wheel, outerpath, or tree. Specifically, in Section 3 we sketch a construction that computes a geometric simultaneous embedding of a wheel and a cycle, which immediately implies an embedding for a wheel and a matching. In Section 4 and 5 we describe algorithms to embed a matching together with two specific types of outerplanar graphs, namely *outerzigzags* and *outerpaths*. An outerzigzag is also known as a triangle strip. Its weak dual is a path and each of its vertices has degree at most 4. An outerpath is simply an outerplanar graph whose weak dual is a path. Our result for outerpaths of course subsumes the result for outerzigzags, but we nevertheless first present the construction for outerzigzags, to introduce our techniques on a conceptually simpler class of graphs. The algorithms for the wheel, the outerzigzag, and the outerpath, preserve the "natural" embedding of these graphs. That is, the center of the wheel is not incident to the outer face, and the embedding of outerplanar graphs is outerplanar. Note here, that an outerplanar graph and a path may not have a geometric simultaneous embedding if the circular ordering of the edges around the vertices of the outerplanar graph is fixed a-priori [7].

In Section 6 we present an algorithm that computes a geometric simultaneous embedding of a tree and a matching. This algorithm is inspired by and closely related to an algorithm by Di Giacomo *et al.* [2]. Since a path can be viewed as two matchings, our proof sheds some new light on the embeddabilty question for a tree and a path.

All our drawing algorithms minimize the number of orientations used to draw the edges of the matching. This may simplify the visual inspection of the data and of their relationships in practice. Consider the simple example in Fig. 1 (right). It immediately shows that a geometric simultaneous embedding of an outerpath or wheel with a matching requires the matching edges to have at least two orientations. Our constructions match this bound. When embedding a matching and a tree, we can even draw all matching edges horizontally.

2 Planar Graph and Matching

Theorem 1. *There exists a planar graph and a matching that do not admit a geometric simultaneous embedding.*

Consider the planar graph (black) and the matching (gray) depicted in Fig. 2. One can argue that either the subgraph induced by the vertices marked with boxes or the subgraph induced by the vertices marked with circles will always incur at least one crossing. The details can be found in the full paper.

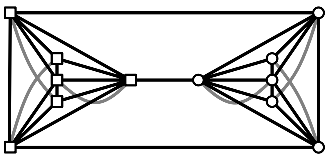

Fig. 2. A planar graph (black) and a matching (gray) that do not admit a geometric simultaneous embedding

3 Wheel and Matching

We can in fact compute a geometric simultaneous embedding of a wheel and a cycle, which immediately implies the result for a wheel and a matching. The construction is sketched in Fig. 3. The center of the wheel is marked with a box, the rim is drawn in black, and the cycle is drawn in gray. When we use this construction for a matching, then all but one matching edges are drawn vertically, the remaining edge (the one edge that is necessarily shared with a spoke of the wheel) is drawn horizontally. Details can be found in the full paper.

Theorem 2. *A wheel and a cycle always admit a geometric simultaneous embedding.*

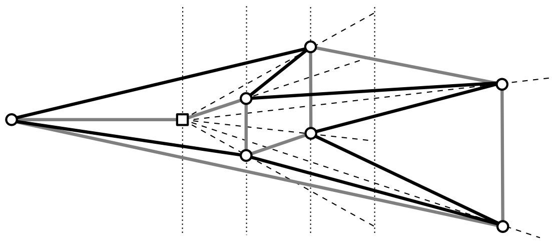

Fig. 3. A geometric simultaneous embedding of a wheel and a cycle

4 Outerzigzag and Matching

Recall that an outerzigzag is a triangle strip: it is a triangulated outerplanar graph, whose weak dual is a path and whose vertices have degree at most 4. More precisely, there are exactly two vertices of degree 2, two vertices of degree 3, and all other vertices have degree 4. Let $G_1 = (V, E_1)$ be an outerzigzag and let $G_2 = (V, E_2)$ be a matching. We first place the vertices of V in such a way, that their placement induces a regular plane drawing of G_1. We then move some of the vertices vertically to planarize G_2, while keeping the drawing of G_1 planar.

Specifically, we place the vertices of V on a grid of size $2n \times 4n$ at positions $(0, 0)$, $(2, 1)$, $(4, 0)$, $(6, 1)$, $(8, 0)$, etc. One of the degree-2 vertices of G_1 is drawn at $(0, 0)$, the remainder is drawn in such a way, that the edges of G_1 always

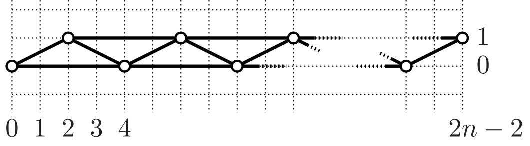

Fig. 4. Drawing an outerzigzag G_1 on a grid

connect two consecutive points on the line $y = 0$, or two consecutive points on the line $y = 1$, or two points at distance $\sqrt{5}$, see Fig. 4.

We classify the edges of the matching G_2 based on the placement of their vertices on the grid:

BB-edges connect any two vertices on the line $y = 0$.
TT-edges connect any two vertices on the line $y = 1$.
BT-edges connect two vertices $(i, 0)$ and $(j, 1)$ with $i < j$.
TB-edges connect two vertices $(i, 1)$ and $(j, 0)$ with $i < j$.

We then move half of the vertices of V vertically according to three simple rules:

1. Only the right vertex of a matching edge moves, the left vertex is fixed.
2. The right vertex of every BT-edge and every TT-edge is moved up until the edge has slope $+1$.
3. The right vertex of every TB-edge and every BB-edge is moved down until the edge has slope -1.

See Fig. 5 for an example. It is easy to see that the displacements preserve the planarity of the embedding of G_1, because vertices only move vertically. The displacements also make G_2 planar: all edges of E_2 with slope $+1$ are on parallel diagonal lines, so they cannot intersect. Symmetrically, all edges of E_2 with slope -1 cannot intersect. Finally, an edge with slope -1 and an edge with slope $+1$ from E_2 cannot intersect because their only y-overlap is between 0 and 1, and here they are sufficiently separated to prevent intersections. Clearly this construction uses only two orientations for the edges of the matching.

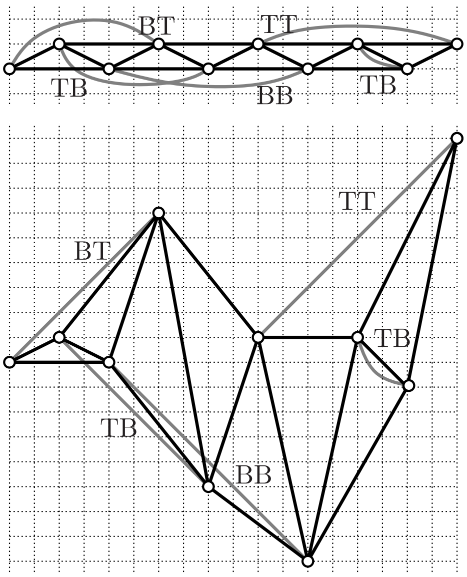

Fig. 5. A geometric simultaneous embedding of an outerzigzag (black) and a matching (gray)

Theorem 3. *An outerzigzag and a matching always admit a geometric simultaneous embedding.*

5 Outerpath and Matching

We now extend the approach for outerzigzags to outerpaths. First, we assume that the outerpath is triangulated. Since a triangulated outerpath has two vertices of degree 2, we can make them the ends of an initial placement. We place all vertices on two horizontal lines $y = 0$ and $y = 1$ in such a way, that we obtain

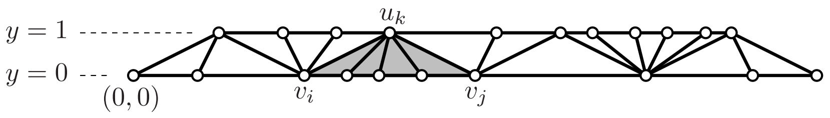

Fig. 6. Outerpath with one fan indicated in gray

a plane drawing of the outerpath G_1 (see Fig. 6). We say that vertices which lie on the line $y = 1$ are on the *top chain*, correspondingly, vertices which lie on the line $y = 0$ are on the *bottom chain*. The leftmost vertex is placed at $(0, 0)$. This initial placement again induces a classification of the edges of the matching into TT-edges, TB-edges, BT-edges, and BB-edges. In the final drawing these edges have slopes of -1 or $+1$ as before. However, the embedding algorithm for outerpaths needs to move vertices not only vertically, but also horizontally and hence the x-order of the vertices in the initial placement is not preserved.

We view an outerpath as a sequence of *maximal fans* that are alternatingly directed upwards and downwards. A maximal fan shares its first and last edge with a neighboring fan. A downward fan is indicated in gray in Fig. 6. We denote by d the maximum degree of any vertex in the outerpath G_1.

Our algorithm works as follows: we treat one fan after the other, moving from left to right. When we treat a fan, we place its vertices at new locations to planarize G_2, while keeping the drawing of G_1 planar. In the following we explain the placement algorithm for a downward fan F, upward fans a treated similarly. We denote the single *apex* vertex of F by u_k and its sequence of *finger* vertices by $v_i, \dots, v_j$ (see Fig. 6). Note that $i < j$; if $i = j$ then the outerpath was not triangulated or F was not maximal. We place all vertices of F, with the exception of v_i and v_j. Vertex v_i has already been placed, since it is the apex of the preceding fan (or it is the leftmost vertex which remains fixed). We do not place v_j since it is the apex of the following fan and will be placed when that fan is treated. We distinguish three cases, depending on the matching partner of the apex u_k. Case (1): the matching partner of u_k has already been placed, Case (2): the matching partner of u_k has not been placed yet and it is not among $v_{i+1}, \dots, v_{j-1}$, and Case (3): the matching partner of u_k is among $v_{i+1}, \dots, v_{j-1}$.

Case (1). Apex u_k has a matching partner that has already been placed. Hence the matching partner lies either on the top chain and has an index smaller than k, or it lies on the bottom chain and has an index smaller than or equal to i. Let X denote the total width (x-extent) of the construction so far. We place u_k at x-coordinate $2X + 1$ and then move u_k upwards until it lies on the line with slope $+1$ through its matching partner (see Fig. 7 (left)).

Next we place $v_{i+1}, \dots, v_{j-1}$ at positions $(2X, 0)$, $(2X+1/d, 0)$, …, $(2X+(j-i-2)/d, 0)$. Consider the $j-i-1$ lines through u_k and each of $v_{i+1}, \dots, v_{j-1}$. If we ensure that the final placements of $v_{i+1}, \dots, v_{j-1}$ lie on these lines, then we will never invert any triangle of the fan. We now move those vertices of $v_{i+1}, \dots, v_{j-1}$ that are right vertices of matching edges down on their lines until they reach the proper position, determined by the slope -1 lines through their matching partners. Those vertices of $v_{i+1}, \dots, v_{j-1}$ that are left vertices of matching edges

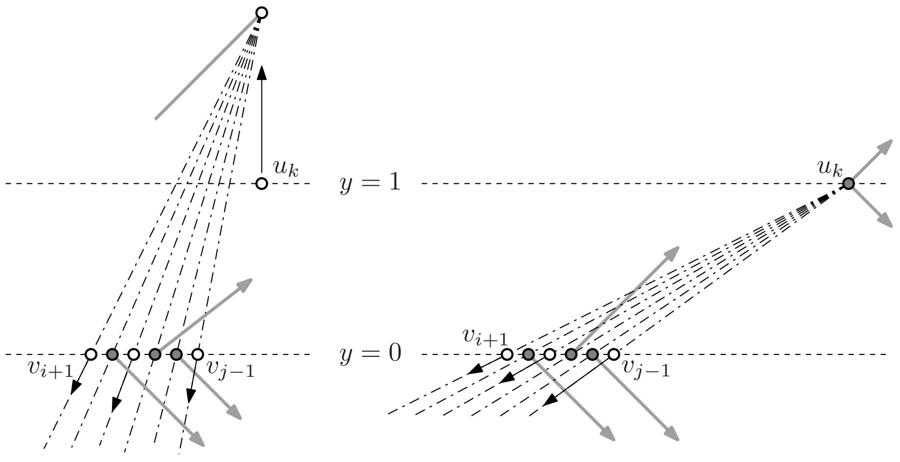

Fig. 7. Left: Case 1, u_k is a right vertex of a matching edge. Right: Case 2, u_k is a left vertex of a matching edge.

stay where they are; they define slope -1 or $+1$ lines on which their matching partners will be placed eventually. By construction, none of these lines intersect to the right of the vertices that defined them, also not with lines defined by vertices treated earlier (see Fig. 7 (left)).

See Fig. 8 for a global sketch of Case 1. Note that v_{i+1} stays to the right of all vertices placed before. This is true because the line defined by u_k and v_{i+1} has slope > 1, and the separation between v_{i+1} and the previously placed vertices is at least X. The value of X also bounds the y-extent for the previously placed vertices to the range $[-X, +X]$, since the edges of the matching have slopes -1 and $+1$. Further note that triangle $\triangle u_k v_{i+1} v_i$ is not inverted, regardless of where v_i is placed in the initial part and whether v_{i+1} is moved on its line. Finally, note that v_j can be placed anywhere on the line $y = 0$ lower, as long as its x-coordinate is at least that of u_k: the triangle $\triangle u_k v_j v_{j-1}$ will not be inverted.

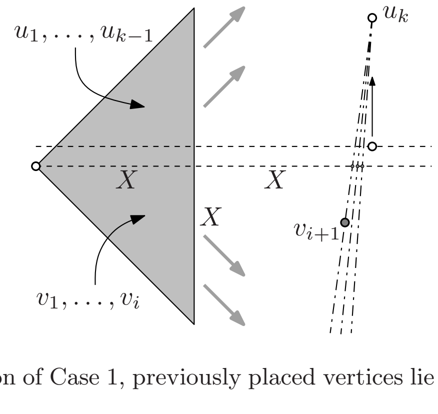

Fig. 8. Global situation of Case 1, previously placed vertices lie inside the gray triangle

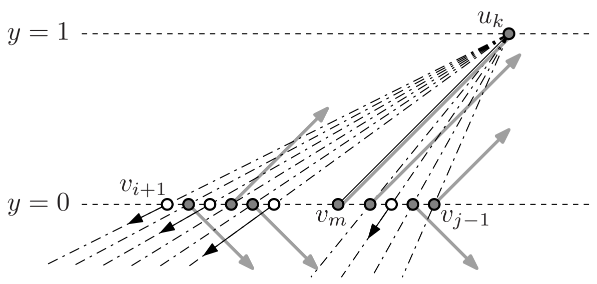

Fig. 9. Case 3

Case (2). Apex u_k has a matching partner that has not been placed yet and which is not among $v_{i+1}, \ldots, v_{j-1}$. We place u_k at position $(3X+2, 1)$, where X is again defined as the total width so far. Next we place the vertices $v_{i+1}, \ldots, v_{j-1}$ at positions $(3X, 0)$, $(3X + 1/d, 0)$, ..., $(3X + (j - i - 2)/d, 0)$. Consider the $j - i - 1$ lines through u_k and each of $v_{i+1}, \ldots, v_{j-1}$. We again move those vertices of $v_{i+1}, \ldots, v_{j-1}$ that are right vertices of matching edges down on their lines until they reach the proper position, determined by the slope -1 lines through their matching partners (see Fig. 7 (right)).

All lines on which the vertices move have slope at least $1/2$, implying that all vertices of $v_{i+1}, \ldots, v_{j-1}$ are placed to the right of all previously placed vertices, due to the x-separation of at least $2X$. Again we note that $\triangle u_k v_{i+1} v_i$ is not inverted, and that v_j may be placed anywhere to the right of u_k without the risk of inverting $\triangle u_k v_j v_{j-1}$.

Case (3). Apex u_k has a matching partner v_m that is among $v_{i+1}, \ldots, v_{j-1}$, see Fig. 9. We place u_k at position $(3X + 2, 1)$ and v_m at position $(3X + 1, 0)$, where X is again the total width so far. Note that the edge (u_k, v_m) is an edge of both G_1 and G_2. Next we place the vertices $v_{i+1}, \ldots, v_{m-1}$ at positions $(3X, 0)$, $(3X + 1/d, 0)$, ..., $(3X + (m - i - 2)/d, 0)$, and the vertices $v_{m+1}, \ldots, v_{j-1}$ at positions $(3X + 1 + 1/d, 0)$, $(3X + 1 + 2/d, 0)$, ..., $(3X + 1 + (j - m - 1)/d, 0)$. As before we now use the lines through u_k and each of $v_{i+1}, \ldots, v_{m-1}, v_{m+1}, \ldots, v_{j-1}$ to move vertices down if they are right vertices of matching edges.

Theorem 4. *An outerpath and a matching always admit a geometric simultaneous embedding.*

6 Tree and Matching

Our algorithm that computes a geometric simultaneous embedding for a tree and a matching is inspired by and closely related to an algorithm by Di Giacomo *et al.* [2], which computes a *matched drawing* of two trees. Matched drawings are a relaxation of geometric simultaneous embeddings. Specifically, two planar graphs G_1 and G_2 are *matched*, if they are defined on two vertex sets V_1 and V_2 of the same cardinality and if there is a one-to-one mapping between V_1

and V_2. A matched drawing of two matched graphs is a pair of planar straight-line drawings, such that matched vertices of G_1 and G_2 are assigned the same y-coordinate. A geometric simultaneous embedding of a tree and a matching is in essence a matched drawing of half of the vertices of the tree with the other half. And indeed, the algorithm by Di Giacomo *et al.* can be adapted in a straightforward manner to compute a geometric simultaneous embedding of a tree and a matching. However, the edges of the matching in the resulting drawing will in general not all have the same orientation. In the remainder of this section we show how to refine the construction from [2], to compute a simultaneous embedding where all matching edges are drawn horizontally.

We place the vertices one by one, always placing the two vertices of a matching edge consecutively at the same y-coordinate. We use y-coordinates $1, \ldots, n/2$, from the outside in. That is, at any point of the construction, there are two indices i and j with $1 \leq i \leq j \leq n/2$ such that the coordinates $1, \ldots, i-1$ and $j+1, \ldots, n/2$ have been used, and the coordinates $i, \ldots, j$ have not been used yet. At every even placement we decide if we should place the next vertex at the top or at the bottom, that is, at the highest or the lowest available y-coordinate.

Let T be the tree with some of its vertices already placed. The placed vertices partition the tree into connected components (subtrees); we call each component up to and including the placed vertices a *rope*. The placed vertices incident to a rope are called the *knots* of that rope. We maintain the following invariant: after every odd placement, every rope of T has one or two knots, but not more. After an even placement this invariant might be false for exactly one rope, which has three knots. There is a unique vertex, which we call the *splitter*, that lies on the three paths between the knots. We show below how to restore the invariant with the next odd placement by choosing the splitter as the next vertex to place.

Since we place vertices from the outside in, there are nine types of ropes which we encounter during the construction. They are the degree-1 ropes with one knot at the top or at the bottom, the degree-2 ropes with two knots at the top, or two at the bottom, or one at the top and one at the bottom, and the degree-3 ropes with zero, one, two, or three knots at the top and three, two, one, or zero knots at the bottom. We call these ropes T-rope, B-rope, TT-rope, BB-rope, TB-rope, BBB-rope, TBB-rope, TTB-rope, or TTT-rope.

Even placement. The invariant above implies that before an even placement there are only degree-1 and degree-2 ropes. Furthermore, there is exactly one edge (v, w) of the matching M that has one, but not both of its vertices placed. We assume that v has been placed and place w next, at the same y-coordinate as v. The exact placement depends on the type of rope w is part of, as well as the y-coordinate of v. Fig. 10 shows the cases for T-ropes, TB-ropes, and TT-ropes, B-ropes and BB-ropes are symmetric. Placing w can create at most two degree-2 ropes or one degree-3 rope, plus zero or more degree-1 ropes. New degree-1 ropes all have w as their knot. The new degree-2 ropes may have no internal vertices, in which case they are fully placed or *tight*, as they are a straight edge. Placing w creates a degree-3 rope, if w was part of a degree-2 rope but did not lie on the path between its two knots. In this case a new splitter s is identified

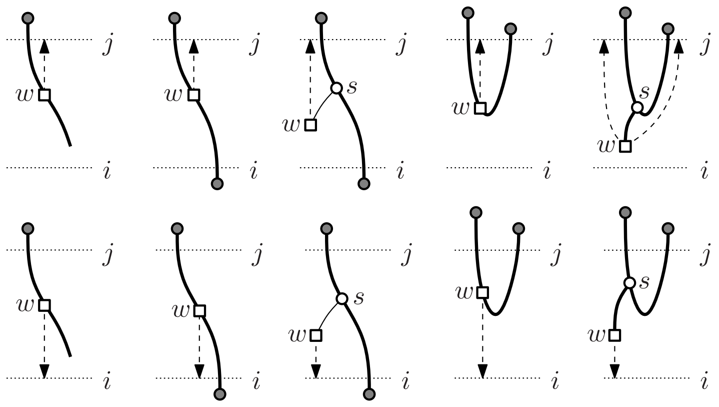

Fig. 10. Even placement for T-ropes, TB-ropes, and TT-ropes

(marked by a circle in Fig. 10), which is placed in the next odd placement. The top right case depicted in Fig. 10 shows two dashed arrows, indicating that there are two possible locations for w. Which of the two we have to use depends on the matching partner of the splitter s. We explain below how to make this decision.

Odd placement. Before an odd placement, all matching edges are either completely placed, or not placed at all. There are two cases: the previous even placement left us with a splitter, or not. If there is no splitter, then we place any unplaced vertex, whose placement does not create a splitter. Any vertex that is directly adjacent to an already placed vertex qualifies. If there is a splitter s, then we place it next. If s is part of a TTT-rope or a TTB-rope, then we place it at the lowest unused y-coordinate i, which creates two or three new TB-ropes and one or zero new BB-ropes. Symmetrically, if s is part of a TBB-rope or a BBB-rope, then we place it at the highest unused y-coordinate j.

There are two additional things to consider. Let u be the matching partner of the splitter s. By construction u has not been placed yet, but it will be placed in the next step, on the same y-coordinate as s.

(1) If s was part of a TTT-rope (or symmetrically, a BBB-rope), then placing s creates three new TB-ropes. If u is part of one of these TB-ropes, then we need to ensure that this particular TB-rope is one of the two "on the outside". The TTT-rope was created by placing a vertex w at y-coordinate j in the previous step (top right case in Fig. 10). Recall that we had two choices for the location of w. One of the two ensures that u is on the outside (see Fig. 11 (top)). Hence we look ahead and place w accordingly. Placing s might also have created one or more B-ropes. If u is part of one of these B-ropes, then we need to ensure again that this particular B-rope is on the outside. We can easily achieve that by ordering the degree 1-ropes with knot s accordingly.

(2) If s was part of a TTB-rope (or symmetrically, a TBB-rope), then placing s creates two TB-ropes and one BB-rope. If u is part of one of the TB-ropes, then we have to ensure again that this particular TB-rope is on the outside.

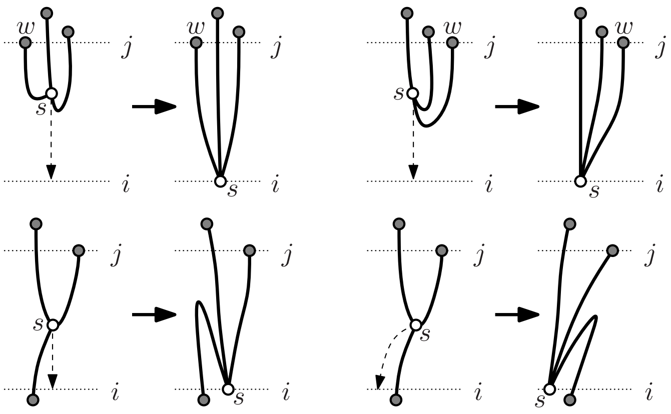

Fig. 11. Odd placement for a splitter and a TTT-rope or a TTB-rope

However, we have a choice of two possible locations for s, see Fig. 11 (bottom). One of the two ensures that u is on the outside, hence we place s accordingly. Again, placing s might also have created one or more B-ropes. In general we place these B-ropes between the two TB-ropes. But if u is part of a B-rope, then we need to place this particular rope on the outside.

Next we have to argue that there is actually space to draw the tree without crossings and with straight edges. For the matching this is obvious since its edges are horizontal and lie on different y-coordinates. We maintain the following invariant, which holds for every rope after every even placement. Let i and j be the lowest and highest unused y-coordinates. There exists a parallelogram between the horizontal lines i and j in which the whole rope can be drawn without crossings and with straight lines. The parallelograms have positive width and have an "alignment" that corresponds to the needs of the rope. In particular, the non-horizontal sides of the parallelogram have a slope s, such that any line with slope s and through a knot of the rope intersects the interior of the parallelogram. Hence every y-coordinate within the parallelogram can be reached by a straight line from its knots. Parallelograms of different ropes are disjoint.

It remains to show how to maintain this invariant as ropes are split. We find the new parallelograms of the sub-ropes inside the parallelogram of the parent. We might have to scale and shear parallelograms to make this work, creating

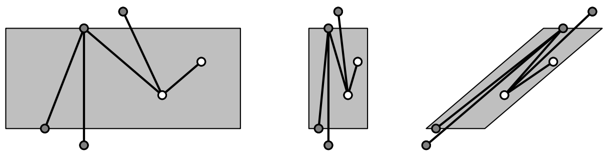

Fig. 12. A parallelogram and its rope can be scaled to become arbitrarily narrow and sheared to get different slopes

extremely narrow parallelograms in the process (see Fig. 12). However, they always keep a strictly positive width. The cases are many, but not difficult and very similar to the ones discussed in [2]. We omit the details.

Theorem 5. *A tree and a matching always admit a geometric simultaneous embedding.*

7 Conclusions

We presented the first results for geometric simultaneous embeddings where one of the graphs is a matching. Specifically, we showed that there exist planar graphs that do not admit a geometric simultaneous embedding with a matching. We do not know whether this negative result holds also under the additional constraint that the matching and the planar graph do not have any edges in common.

We also described algorithms that compute a geometric simultaneous embedding of a matching and a wheel, outerpath, or tree. Our drawing algorithms minimize the number of orientations used to draw the edges of the matching. The main remaining open question is: do an outerplanar graph and a matching always admit a geometric simultaneous embedding?

References

1. Braß, P., Cenek, E., Duncan, C., Efrat, A., Erten, C., Ismailescu, D., Kobourov, S., Lubiw, A., Mitchell, J.: On simultaneous planar graph embeddings. Computational Geometry: Theory & Applications 36(2), 117–130 (2007)
2. Di Giacomo, E., Didimo, W., van Kreveld, M., Liotta, G., Speckmann, B.: Matched drawings of planar graphs. In: Hong, S.-H., Nishizeki, T., Quan, W. (eds.) GD 2007. LNCS, vol. 4875, pp. 183–194. Springer, Heidelberg (2008)
3. Erten, C., Harding, P., Kobourov, S., Wampler, K., Yee, G.: Exploring the computing literature using temporal graph visualization. In: Proc. Conference on Visualization and Data Analysis, pp. 45–56 (2003)
4. Erten, C., Harding, P., Kobourov, S., Wampler, K., Yee, G.: GraphAEL: Graph animations with evolving layouts. In: Liotta, G. (ed.) GD 2003. LNCS, vol. 2912, pp. 98–110. Springer, Heidelberg (2004)
5. Erten, C., Kobourov, S.: Simultaneous embedding of planar graphs with few bends. Journal of Graph Algorithms and Applications 9(3), 347–364 (2005)
6. Estrella-Balderrama, A., Gassner, E., Jünger, M., Percan, M., Schaefer, M., Schulz, M.: Simultaneous geometric graph embeddings. In: Hong, S.-H., Nishizeki, T., Quan, W. (eds.) GD 2007. LNCS, vol. 4875, pp. 280–290. Springer, Heidelberg (2008)
7. Frati, F., Kaufmann, M., Kobourov, S.: Constrained simultaneous and near-simultaneous embeddings. In: Hong, S.-H., Nishizeki, T., Quan, W. (eds.) GD 2007. LNCS, vol. 4875, pp. 268–279. Springer, Heidelberg (2008)
8. Geyer, M., Kaufmann, M., Vrt'o, I.: Two trees which are self-intersecting when drawn simultaneously. Discrete Mathematics (to appear)
9. Ware, C., Purchase, H., Colpoys, L., McGill, M.: Cognitive measurements of graph aesthetics. Information Visualization 1(2), 103–110 (2002)