Drawing Non-Planar Graphs with Crossing-Free Subgraphs*

Patrizio Angelini[1], Carla Binucci[2], Giordano Da Lozzo[1], Walter Didimo[2], Luca Grilli[2], Fabrizio Montecchiani[2], Maurizio Patrignani[1], and Ioannis G. Tollis[3]

[1] Università Roma Tre, Italy
{angelini,dalozzo,patrigna}@dia.uniroma3.it
[2] Università degli Studi di Perugia, Italy
{binucci,didimo,grilli,montecchiani}@diei.unipg.it
[3] Univ. of Crete and Institute of Computer Science-FORTH, Greece
tollis@ics.forth.gr

Abstract. We initiate the study of the following problem: *Given a non-planar graph G and a planar subgraph S of G, does there exist a straight-line drawing Γ of G in the plane such that the edges of S are not crossed in Γ?* We give positive and negative results for different kinds of spanning subgraphs S of G. Moreover, in order to enlarge the subset of instances that admit a solution, we consider the possibility of bending the edges of $G \setminus S$; in this setting different trade-offs between number of bends and drawing area are given.

1 Introduction

Lots of papers in graph drawing address the problem of computing drawings of non-planar graphs with the goal of mitigating the negative effect that edge crossings have on the drawing readability. Many of these papers describe crossing minimization methods, which are effective and computationally feasible for relatively small and sparse graphs (see [8] for a survey). Other papers study which non-planar graphs can be drawn such that the "crossing complexity" of the drawing is somewhat controlled, either in the number or in the type of crossings. They include the study of *k-planar drawings*, in which each edge is crossed at most k times (see, e.g., [7,11,12,15,16,20,24]), of *k-quasi planar drawings*, in which no k pairwise crossing edges exist (see, e.g., [1,2,10,23,26,28]), and of *large angle crossing drawings*, in which any two crossing edges form a sufficiently large angle (see [14] for a survey). Most of these drawings exist only for sparse graphs.

In this paper we initiate the study of a new graph drawing problem concerned with the drawing of non-planar graphs. Namely: *Given a non-planar graph G and a planar subgraph S of G, decide whether G admits a drawing Γ such that the edges of S are not crossed in Γ, and compute Γ if it exists.*

Besides its intrinsic theoretical interest, this problem is also of practical relevance in many application domains. Indeed, distinct groups of edges in a graph may have different semantics, and a group can be more important than another for some applications;

* Work on these results began at the 8th Bertinoro Workshop on Graph drawing. Discussion with other participants is gratefully acknowledged. Part of the research was conducted in the framework of ESF project 10-EuroGIGA-OP-003 GraDR "Graph Drawings and Representations".

in this case a visual interface might attempt to display more important edges in a planar way. Again, the user could benefit from a layout in which a spanning connected subgraph is drawn crossing free, since it would support the user to quickly recognize paths between any two vertices, while keeping the other edges of the graph visible.

We remark that the problem of recognizing specific types of subgraphs that are not self-crossing (or that have few crossings) in a given drawing Γ, has been previously studied (see, e.g., [17,19,22,25]). This problem, which turns out to be NP-hard for most different kinds of instances, is also very different from our problem. Indeed, in our setting the drawing is not the input, but the output of the problem. Also, we require that the given subgraph S is not crossed by any edge of the graph, not only by its own edges.

In this paper we concentrate on the case in which S is a spanning subgraph of G and consider both straight-line and polyline drawings of G. Namely:

(i) In the straight-line drawing setting we prove that if S is any given spanning spider or caterpillar, then a drawing of G where S is crossing free always exists; such a drawing can be computed in linear time and requires polynomial area (Section 3.1). We also show that this positive result cannot be extended to any spanning tree, but we describe a large family of spanning trees that always admit a solution, and we show that any graph G contains such a spanning tree; unfortunately, our drawing technique for trees may require exponential area. Finally, we characterize the instances $\langle G, S\rangle$ that admit a solution when S is a spanning triconnected subgraph, and we provide a polynomial-time testing and drawing algorithm, whose layouts have polynomial area (Section 3.2).

(ii) We investigate polyline drawings where only the edges of $G \setminus S$ are allowed to bend. In this setting, we show that all spanning trees can be realized without crossings in a drawing of G of polynomial area, and we describe efficient algorithms that provide different trade-offs between number of bends per edge and drawing area (Section 4). Also, in Section 5 we briefly discuss a characterization of the instances $\langle G, S\rangle$ that admit a drawing when S is any given biconnected spanning subgraph.

Due to space restrictions, some proofs are omitted or only sketched in the text; full proofs for all results can be found in the [4].

2 Preliminaries and Definitions

We assume familiarity with basic concepts of graph drawing and planarity (see, e.g., [9]). Let $G(V, E)$ be a graph and let Γ be a drawing of G in the plane. If all vertices and edge bends of Γ have integer coordinates, then Γ is an *integer grid drawing* of G, and the *area* of Γ is the area of the minimum bounding box of Γ. Otherwise, suppose that Γ is not an integer grid drawing and let d_{min} be the minimum distance between two points of Γ on which either vertices or bends are drawn. In this case, the *area* of Γ is defined as the area of the minimum bounding box of a drawing obtained by scaling Γ by a constant c such that $c \times d_{min} = 1$; this corresponds to establish a certain resolution rule between vertices and bends of Γ, which is comparable to that of an integer grid drawing.

Let $G(V, E)$ be a graph and let $S(V, W)$, $W \subseteq E$, be a spanning subgraph of G. A straight-line drawing Γ of G such that S is crossing-free in Γ (i.e., such that crossings occur only between edges of $E \setminus W$) is called a *straight-line compatible drawing* of $\langle G, S\rangle$. If each edge of $E \setminus W$ has at most k bends in Γ (but still the subdrawing of S is straight-line and crossing-free), Γ is called a *k-bend compatible drawing* of $\langle G, S\rangle$.

If S is a rooted spanning tree of G such that every edge of $G \setminus S$ connects either vertices at the same level of S or vertices that are on consecutive levels, then we say that S is a *BFS-tree* of G.

A *star* is a tree $T(V, E)$ such that all its vertices but one have degree one, that is, $V = \{u, v_1, v_2, \dots, v_k\}$ and $E = \{(u, v_1), (u, v_2), \dots, (u, v_k)\}$; any subdivision of T (including T), is a *spider*: vertex u is the *center* of the spider and each path from u to v_i is a *leg* of the spider. A *caterpillar* is a tree such that removing all its leaves (and their incident edges) results in a path, which is called the *spine* of the caterpillar. The one-degree vertices attached to a spine vertex v are called the *leaves* of v.

In the remainder of the paper we implicitly assume that G is always a connected graph (if the graph is not connected, our results apply for any connected component).

3 Straight-Line Drawings

We start studying straight-line compatible drawings of pairs $\langle G, S\rangle$: Section 3.1 concentrates on the case in which S is a spanning tree, while Section 3.2 investigates the case in which S is a spanning triconnected graph.

3.1 Spanning Trees

The simplest case is when S is a given Hamiltonian path of G; in this case Γ can be easily computed by drawing all vertices of S in convex position, according to the ordering they occur in the path. In the following we prove that in fact a straight-line compatible drawing Γ of $\langle G, S\rangle$ can be always constructed in the more general cases in which S is a spanning spider (Theorem 1), or a spanning caterpillar (Theorem 2), or a BFS-tree (Theorem 3); our construction techniques guarantee polynomial-area drawings for spiders and caterpillars, while require exponential area for BFS-trees. On the negative side, we show that if S is an arbitrary spanning tree, a straight-line compatible drawing of $\langle G, S\rangle$ may not exist (Lemmas 1 and 2).

Theorem 1. *Let G be a graph with n vertices and m edges, and let S be a spanning spider of G. There exists an integer grid straight-line compatible drawing Γ of $\langle G, S\rangle$. Drawing Γ can be computed in $O(n + m)$ time and has $O(n^3)$ area.*

Proof. Let u be the center of S and let $\pi_1, \pi_2, \dots, \pi_k$ be the legs of S. Also, denote by v_i the vertex of degree one of leg π_i $(1 \leq i \leq k)$. Order the vertices of S distinct from u such that: (i) the vertices of each π_i are ordered in the same way they appear in the simple path of S from u to v_i; (ii) the vertices of π_i precede those of π_{i+1} $(1 \leq i \leq k - 1)$. If v is the vertex at position j $(0 \leq j \leq n - 2)$ in the ordering defined above, draw v at coordinates (j^2, j). Finally, draw u at coordinates $(0, n - 2)$. With this strategy, all vertices of S are in convex position, and they are all visible from u in such a way that no edge incident to u can cross other edges of Γ. Hence, the edges of S do not cross other edges in Γ. The area of Γ is $(n - 2)^2 \times (n - 2) = O(n^3)$ and Γ is constructed in linear time. □

The next algorithm computes a straight-line compatible drawing of $\langle G, S \rangle$ when S is a spanning caterpillar. Theorem 2 proves its correctness, time and area requirements. Although the drawing area is still polynomial, the layout is not an integer grid drawing.

Algorithm. STRAIGHT-LINE-CATERPILLAR. Denote by $u_1, u_2, \ldots, u_k$ the vertices of the spine of S. Also, for each spine vertex u_i $(1 \leq i \leq k)$, let $v_{i1}, \ldots, v_{in_i}$ be its leaves in S (refer to the bottom image in Fig. 1(a)). The algorithm temporarily adds to S and G some dummy vertices, which will be removed in the final drawing. Namely, for each u_i, it attaches to u_i two dummy leaves, s_i and t_i. Also, it adds a dummy spine vertex u_{k+1} attached to u_k and a dummy leaf s_{k+1} of u_{k+1} (see the top image in Fig. 1(a)). Call G' and S' the new graph and the new caterpillar obtained by augmenting G and S with these dummy vertices.

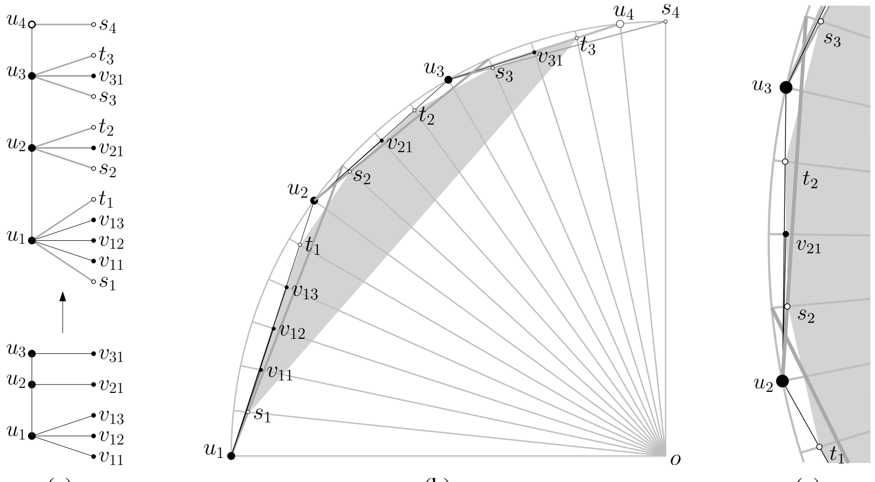

Fig. 1. Illustration of Algorithm STRAIGHT-LINE-CATERPILLAR: (a) a caterpillar S and its augmented version S'; (b) a drawing of S'; edges of the graph connecting leaves of S are drawn in the gray (convex) region; (c) enlarged detail of the picture (b)

The construction of a drawing Γ' of G' is illustrated in Fig. 1(b). Consider a quarter of circumference C with center o and radius r. Let N be the total number of vertices of G'. Let $\{p_1, p_2, \ldots, p_N\}$ be N equally spaced points along C in clockwise order, where $\overline{op_1}$ and $\overline{op_N}$ are a horizontal and a vertical segment, respectively. For each $1 \leq i \leq k$, consider the ordered list of vertices $L_i = \{u_i, s_i, v_{i1}, \ldots v_{in_i}, t_i\}$, and let L be the concatenation of all L_i. Also, append to L the vertices u_{k+1} and s_{k+1}, in this order. Clearly the number of vertices in L equals N. For a vertex $v \in L$, denote by $j(v)$ the position of v in L. Vertex u_i is drawn at point $p_{j(u_i)}$ $(1 \leq i \leq k)$; also, vertices u_{k+1} and s_{k+1} are drawn at points p_{N-1} and p_N, respectively. Each leaf v of S' will be suitably drawn along radius $\overline{op_{j(v)}}$ of C. More precisely, for any $i \in \{1, \ldots, k\}$, let a_i be the intersection point between segments $\overline{p_{j(u_i)}p_{j(s_{i+1})}}$ and $\overline{op_{j(s_i)}}$, and let b_i be the intersection point between segments $\overline{p_{j(u_i)}p_{j(u_{i+1})}}$ and $\overline{op_{j(t_i)}}$. Vertices s_i and

t_i are drawn at points a_i and b_i, respectively. Also, let A_i be the circular arc that is tangent to $\overline{p_{j(u_i)}p_{j(u_{i+1})}}$ at point b_i, and that passes through a_i; vertex v_{ih} is drawn at the intersection point between A_i and $\overline{op_{j(v_{ih})}}$ $(1 \leq h \leq n_i)$.

Once all vertices of G' are drawn, each edge of G' is drawn in Γ' as a straight-line segment between its end-vertices. Drawing Γ is obtained from Γ' by deleting all dummy vertices and their incident edges.

Theorem 2. *Let G be graph with n vertices and m edges, and let S be a spanning caterpillar of G. There exists a straight-line compatible drawing Γ of $\langle G, S\rangle$. Drawing Γ can be computed in $O(n+m)$ time in the real RAM model*[1] *and has $O(n^2)$ area.*

Proof sketch: Let Γ be the output of Algorithm STRAIGHT-LINE-CATERPILLAR. We first prove that Γ is a straight-line compatible drawing of $\langle G, S\rangle$, and then we analyze time complexity and area requirement. We adopt the same notation used in the description of the algorithm.

CORRECTNESS. We have to prove that in Γ the edges of S are never crossed. Our construction places all spine vertices of S' (and hence of S) in convex position. It is also possible to see that the leaves of S' are all in convex position and form a convex polygon P. Since by construction the edges of S are all outside P in Γ, these edges cannot be crossed by edges of G connecting two leaves of S. Also, it is immediate to see that an edge of S cannot be crossed by another edge of S and it is not difficult to see that an edge of S cannot be crossed by an edge of G connecting either two non-consecutive spine vertices or a leaf of S to a spine vertex of S.

TIME AND AREA REQUIREMENT. Clearly, the construction of Γ' (and then of Γ) can be executed in linear time, in the real RAM model. About the area, let $d_{\min}$ be the minimum distance between any two vertices of Γ. It can be proved that if we require $d_{\min} \geq 1$ then $r < \frac{\sqrt{2}}{\beta}$, for $\beta = \theta(\frac{1}{N})$. Thus, the area of Γ is $O(N^2) = O(n^2)$. □

The next lemmas show that, unfortunately, Theorem 1 and Theorem 2 cannot be extended to any spanning tree S, that is, there are pairs $\langle G, S\rangle$ that do not admit a straight-line compatible drawing, even if S is a ternary or a binary tree.

Lemma 1. *Let G be the complete graph on 13 vertices and let S be a complete rooted ternary tree that spans G. There is no straight-line compatible drawing of $\langle G, S\rangle$.*

Proof sketch: Suppose, for a contradiction, that a straight-line compatible drawing Γ of $\langle G, S\rangle$ exists. Let r be the root of S (see Fig. 2(a)). Note that r is the only vertex of S with degree 3. Let u, v, w be the three neighbors of r in S. Two are the cases: either one of u, v, w (say u) lies inside triangle $\triangle(r, v, w)$ (Case 1, see Fig. 2(b)); or r lies inside triangle $\triangle(u, v, w)$ (Case 2).

In Case 1, consider a child u_1 of u. Vertex u_1 is placed in Γ in such a way that u lies inside either triangle $\triangle(u_1, r, w)$ or triangle $\triangle(u_1, r, v)$; assume the former (see Fig. 2(c)). Then, consider another child u_2 of u; in order for edge (u, u_2) not to cross any edge, also u_2 has to lie inside $\triangle(u_1, r, w)$, in such a way that both u and u_1 lie inside triangle $\triangle(u_2, r, v)$. This implies that u lies inside $\triangle(u_1, r, u_2)$ (see Fig. 2(d)),

[1] We also assume that basic trigonometric functions are executed in constant time.

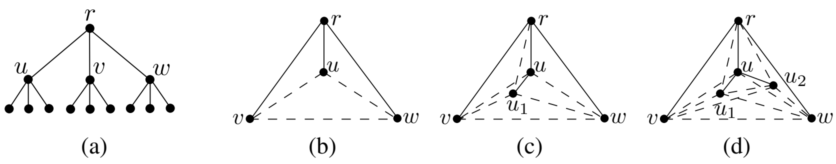

Fig. 2. Illustration for Lemma 1: (a) A complete rooted ternary tree with 13 vertices. (b) Case 1 in the proof; u lies inside $\triangle(r, v, w)$. (c) Placement of u_1. (d) Placement of u_2.

together with its last child u_3. However, u_3 cannot be placed in any of the three triangles in which $\triangle(u_1, r, u_2)$ is partitioned by the edges (of S) connecting u to u_1, to r, and to u_2, respectively, without introducing any crossing involving edges of S, a contradiction. Case 2 can be analyzed with analogous considerations. □

The proof strategy of Lemma 2 is similar to that of Lemma 1.

Lemma 2. *Let G be the complete graph on 22 vertices and let S be a complete unrooted binary tree that spans G. There is no straight-line compatible drawing of $\langle G, S\rangle$.*

In the light of Lemmas 1 and 2, it is natural to ask whether there are specific subfamilies of spanning trees S (other than paths, spiders, and caterpillars) such that a straight-line compatible drawing of $\langle G, S\rangle$ always exists. The next algorithm gives a positive answer to this question: it computes a straight-line compatible drawing when S is a BFS-tree of G. Theorem 3 proves the algorithm correctness, its time complexity, and its area requirement.

Algorithm. STRAIGHT-LINE-BFS-TREE. Let u be the root of S (which is at level 0) and let $u_{l1}, \dots, u_{lk_l}$ be the vertices at level $l \in \{1, \dots, d\}$, where d is the depth of S. The algorithm temporarily adds to S and G some dummy vertices, which will be removed in the final drawing. Namely, for each u_{li}, $1 \le l \le d-1$ and $1 \le i \le k_l$, it attaches to u_{li} one more (leftmost) child s_{li}. Also, it attaches to root u a dummy (rightmost) child t. Denote by G' and S' the new graph and the new tree, respectively. Notice that S' is still a BFS-tree of G'. The algorithm iteratively computes a drawing Γ' of G'. For $l = 1, \dots, d$, the algorithm defines a circumference C_l with center $o = (0, 0)$ and radius $r_l < r_{l-1}$ ($C_1, \dots, C_d$ are concentric). The vertices of level l are drawn on the quarter of C_l going from point $(-r_l, 0)$ to point $(0, r_l)$ clockwise.

Let $\{u_{11}, \dots, u_{1k_1}, t\}$ be the ordered list of the children of root u and let $\{p_{11}, \dots, p_{1k_1}, p_t\}$ be $k_1 + 1$ equally spaced points along C_1 in clockwise order, where $\overline{op_{11}}$ and $\overline{op_t}$ are a horizontal and a vertical segment, respectively. Vertex u_{1j} is drawn on p_{1j} ($1 \le j \le k_1$) and vertex t is drawn on p_t. Also, u is drawn on point $(-r_1, r_1)$.

Assume now that all vertices $u_{l1}, \dots, u_{lk_l}$ of level l have been drawn ($1 \le l \le d-1$) in this order on the sequence of points $\{q_1, \dots, q_{k_l}\}$, along C_l. The algorithm draws the vertices of level $l+1$ as follows. Let $\overline{q_i q_{i+1}}$ be the chords of C_l, for $1 \le i \le k_l - 1$, and let c_l be the shortest of these chords. The radius r_{l+1} of C_{l+1} is chosen arbitrarily in such a way that C_{l+1} intersects c_l in two points and $r_{l+1} < r_l$. This implies that C_{l+1} also intersects every chord $\overline{q_i q_{i+1}}$ in two points. For $1 \le i \le k_l$, denote by $L(u_{li})$ = $\{v_1, \dots, v_{n_{li}}\}$ the ordered list of children of u_{li} in G'. Also, let a_i be the intersection point between $\overline{q_i q_{i+1}}$ and C_{l+1} that is closest to q_i, and let ℓ_i be the line through q_i tangent to C_{l+1}; denote by b_i the tangent point between ℓ_i and C_{l+1}. Let A_{l+1} be the arc of C_{l+1} between a_i and b_i, and let $\{p_0, p_1, \ldots, p_{n_{li}}\}$ be $n_{li} + 1$ equally spaced points along A_{l+1} in clockwise order. For $v \in L(u_{li})$, denote by $j(v)$ the position of v in $L(u_{li})$. Vertex v_j is drawn on $p_{j(v_j)}$ $(1 \leq j \leq n_{li})$ and vertex s_{li} is drawn on p_0.

Once all vertices of G' are drawn each edge of G' is drawn in Γ' as a straight-line segment between its end-vertices. Drawing Γ is obtained from Γ' by deleting all dummy vertices and their incident edges.

Theorem 3. *Let G be a graph with n vertices and m edges, and let S be a BFS-tree of G. There exists a straight-line compatible drawing Γ of $\langle G, S\rangle$. Drawing Γ can be computed in $O(n + m)$ time in the real RAM model.*

It is worth observing that any graph G admits a BFS-tree rooted at an arbitrarily chosen vertex r of G. Thus, each graph admits a straight-line drawing Γ such that one of its spanning trees S is never crossed in Γ. Unfortunately, the compatible drawing computed by Algorithm STRAIGHT-LINE-BFS-TREE may require area $\Omega(2^n)$.

3.2 Spanning Triconnected Subgraphs

Here we focus on triconnected spanning subgraph S of G. Clearly, since every tree can be augmented with edges to become a triconnected graph, Lemmas 1 and 2 imply that, if S is a triconnected graph, a straight-line compatible drawing of $\langle G, S\rangle$ may not exist. The next theorem characterizes those instances for which such a drawing exists.

Theorem 4. *Let $G(V, E)$ be a graph, $S(V, W)$ be a spanning planar triconnected subgraph of G, and $\mathcal{E}$ be the unique planar (combinatorial) embedding of S (up to a flip). A straight-line compatible drawing Γ of $\langle G, S\rangle$ exists if and only if: (1) Each edge $e \in E \setminus W$ connects two vertices belonging to the same face of $\mathcal{E}$. (2) There exists a face f of $\mathcal{E}$ containing three vertices such that any pair u, v of them does not separate in the circular order of f the end-vertices $x, y \in f$ of any other edge in $E \setminus W$.*

Proof sketch: Since $\mathcal{E}$ is unique the necessity of Condition 1 is trivial. Its sufficiency would be also trivial if S admitted a convex drawing where the external face is a triangle. Otherwise, suppose that v_1, v_2, and v_3 are three vertices of a face f satisfying Condition 2. Dummy vertices can be added to S among v_1, v_2, and v_3 in order to have a triangular face to be used as external face when computing the convex drawing (for example, using the algorithm in [27]). The necessity of Condition 2 follows from considering any three vertices on the convex hull of a compatible drawing of $\langle G, S\rangle$. □

The next algorithm exploits Theorem 4 in order to decide in polynomial time whether $\langle G, S\rangle$ admits a straight-line compatible drawing.

Algorithm. STRAIGHT-LINE-TRICONNECTED. Let $\mathcal{E}$ be the unique planar embedding of S (up to a flip). The algorithm verifies that each edge of $E \setminus W$ satisfies Condition 1 of Theorem 4 and that there exists a face f of $\mathcal{E}$ containing three vertices v_1, v_2, and v_3, that satisfy Condition 2 of Theorem 4. If both conditions hold, then v_1, v_2, and v_3 can be used to find a straight-line compatible drawing Γ of $\langle G, S\rangle$ as described in the proof of Theorem 4.

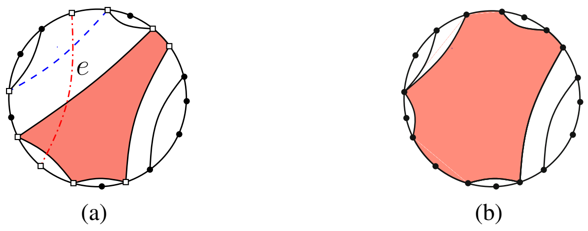

Fig. 3. Two consecutive steps of Algorithm STRAIGHT-LINE-TRICONNECTED. (a) The outerplane graph G_f; the shaded face is `full` (the others are `empty`); the dash-dot edge e is the next edge of E_f to be considered; edges in E_χ are drawn as dashed lines; white squares are vertices of V_χ. (b) Graph G_f after the update due to edge e.

Condition 1 is verified as follows. Construct an auxiliary graph S' from S by subdividing each edge e of W with a dummy vertex v_e. Also, for each face f of $\mathcal{E}$ add to S' a vertex v_f and connect v_f to all non-dummy vertices of f. We have that two vertices of V belong to the same face of $\mathcal{E}$ if and only if their distance in S' is two.

To test Condition 2 of Theorem 4 we perform the following procedure on each face f of $\mathcal{E}$, restricting our attention to the set E_f of edges in $E \setminus W$ whose end-vertices belong to f. We maintain an auxiliary outerplane graph G_f whose vertices are the vertices V_f of f. Each internal face of G_f is either marked as `full` or as `empty`. Faces marked `full` are not adjacent to each other. Intuitively, we have that any three vertices of an `empty` face satisfy Condition 2, while all triples of vertices of a `full` face do not. We initialize G_f with the cycle composed of the vertices and the edges of f and mark its unique internal face as `empty`. At each step an edge e of E_f is considered and G_f is updated. If adding e to G_f splits a single face marked `empty`, we update G_f by splitting such a face into two `empty` faces. If the end-vertices of e belong to a single face marked `full`, we ignore e. Otherwise, adding e to G_f would cross several edges and faces (see Fig. 3(a)). Consider the set E_χ of internal edges of G_f crossed by e. Define a set of vertices V_χ of G_f with the end-vertices of e, the end-vertices of edges of E_χ that are incident to two `empty` faces, the vertices of the `full` faces traversed by e. Remove all edges in E_χ from G_f. Mark the face f' obtained by such a removal as `empty`. Form a new face f_χ inside f' with all vertices in V_χ by connecting them as they appear in the circular order of f, and mark f_χ as `full` (see Fig. 3(b)).

When all the edges of E_f have been considered, if G_f has an internal face marked as `empty`, any three vertices of this face satisfy Condition 2. Else, G_f has a single internal face marked `full` and all triples of vertices of f do not satisfy Condition 2.

Theorem 5. *Let $G(V, E)$ be a graph and let $S(V, W)$ be a spanning triconnected planar subgraph of G. There exists an $O(|V| \times |E \setminus W|)$-time algorithm that decides whether $\langle G, S \rangle$ admits a straight-line compatible drawing Γ and, in the positive case, computes it on an $O(|V|^2) \times O(|V|^2)$ grid.*

Proof sketch: Algorithm STRAIGHT-LINE-TRICONNECTED constructs Γ. Its correctness trivially descends from Theorem 4. Regarding the time complexity, the unique planar embedding $\mathcal{E}$ of S can be computed in $O(|V|)$ time. The auxiliary graph S' for

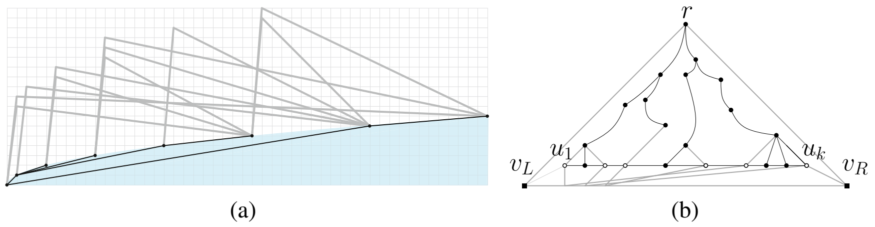

Fig. 4. Illustration of: (a) Algorithm ONE-BEND TREE and (b) Algorithm THREE-BEND TREE; a graph G with a given spanning tree S (black edges)

testing Condition 1 can be constructed in time linear in the size of S. Since S' is a planar graph, deciding if two vertices have distance two can be done in constant time [21]. Thus, testing Condition 1 for all edges in $E \setminus W$ can be done in $O(|V| + |E \setminus W|)$ time. While verifying Condition 1, E_f can be computed, for each face f of $\mathcal{E}$, in $O(|V| + |E \setminus W|)$ time. Since for each face f of $\mathcal{E}$, the size of G_f is $O(|V_f|)$, adding edges in E_f has time complexity $O(|E_f| \times |V_f|)$. Overall, we have that the time complexity of testing Condition 2 is $O(|E \setminus W| \times |V|)$, which gives the time complexity of the whole algorithm. Regarding the area, the algorithm in [5] can be used to obtain in linear time a straight-line grid drawing of S on an $O(|V|^2) \times O(|V|^2)$ grid; this drawing is strictly convex. □

4 Polyline Drawings

We now prove that, using bends along the edges of $G \setminus S$ allows us to compute compatible drawings of pairs $\langle G, S \rangle$ for every spanning tree S of G; such drawings are on a polynomial-area grid. In particular, since edge bends are negatively correlated to the drawing readability, we want to compute k-bend compatible drawings for small values of k. We provide algorithms that offer different trade-offs between number of bends and drawing area. In Section 5 we briefly discuss some preliminary results about 1-bend compatible drawings of $\langle G, S \rangle$ when S is a biconnected spanning subgraph.

Let $G(V, E)$ be a graph with n vertices and m edges, and let $S(V, W)$ be any spanning tree of G. We denote by $x(v)$ and $y(v)$ the x- and the y-coordinate of a vertex v, respectively. The next algorithm computes a 1-bend compatible drawing of $\langle G, S \rangle$.

Algorithm. ONE-BEND TREE. The algorithm works in two steps (refer to Fig. 4(a)).

STEP 1: Consider a point set of size n such that for each point p_i, the x- and y-coordinates of p_i are i^2 and i, respectively. Construct a straight-line drawing of S by placing the vertices on points p_i, $1 \leq i \leq n$, according to a DFS traversal.

STEP 2: Let v_i be the vertex placed on point p_i. For each $i \in \{1, \ldots, n\}$, draw each edge $(v_i, v_j) \in E \setminus W$ such that $j > i$ as a polyline connecting p_i and p_j, and bending at point $(i^2 + 1, n + c)$ where c is a progressive counter, initially set to one.

Theorem 6. *Let $G(V, E)$ be a graph with n vertices and m edges, and let $S(V, W)$ be any spanning tree of G. There exists a 1-bend compatible drawing Γ of $\langle G, S \rangle$. Drawing Γ can be computed in $O(n + m)$ time and has $O(n^2(n + m))$ area.*

Proof sketch: The algorithm that computes Γ is Algorithm ONE-BEND TREE. Note that the drawing of S contained in Γ is planar, and that the edges in $E \setminus W$ are drawn outside the convex region containing the drawing of S. About area requirements, the width of Γ is $O(n^2)$, by construction, while the height of Γ is given by the y-coordinate of the topmost bend point, that is $n + m$. □

Next, we describe an algorithm that constructs 3-bend compatible drawings of pairs $\langle G, S \rangle$ with better area bounds than Algorithm ONE-BEND TREE for sparse graphs.

Algorithm. THREE-BEND TREE. The algorithm works in four steps (see Fig. 4(b)).

STEP 1: Let G' be the graph obtained from G by subdividing each edge $(v_i, v_j) \in E \setminus W$ with two dummy vertices $d_{i,j}$ and $d_{j,i}$. Let S' be the spanning tree of G', rooted at any non-dummy vertex r, obtained by deleting all edges connecting two dummy vertices. Clearly, every dummy vertex is a leaf of S'.

STEP 2: For each vertex of S', order its children arbitrarily, thus inducing a left-to-right order of the leaves of S'. Rename the leaves of S' as $u_1, \dots, u_k$ following this order. For each $i \in \{1, \dots, k-1\}$, add an edge (u_i, u_{i+1}) to S'. Also, add to S' two dummy vertices v_L and v_R, and edges (v_L, r),(v_R, r),(v_L, u_1),(u_k, v_R), (v_L, v_R).

STEP 3: Construct a straight-line grid drawing Γ' of S', as described in [18], in which edge (v_L, v_R) is drawn as a horizontal segment on the outer face, vertices $u_1, \dots, u_k$ all lie on points having the same y-coordinate Y, and the rest of S' is drawn above such points. Remove from Γ' the vertices and edges added in STEP 2.

STEP 4: Compute a drawing Γ of G such that each edge in W is drawn as in Γ', while each edge $(v_i, v_j) \in E \setminus W$ is drawn as a polyline connecting v_i and v_j, bending at $d_{i,j}$, at $d^{j,i}$, and at a point $(c, Y-1)$ where c is a progressive counter, initially set to $x(u_1)$.

Theorem 7. *Let $G(V,E)$ be a graph with n vertices and m edges, and let $S(V,W)$ be any spanning tree of G. There exists a 3-bend compatible drawing Γ of $\langle G, S \rangle$. Drawing Γ can be computed in $O(n+m)$ time and has $O((n+m)^2)$ area.*

Proof sketch: The algorithm that computes Γ is Algorithm THREE-BEND TREE. The drawing of S contained in Γ is planar ([18]) and lies above the horizontal line $y = Y$. The area bounds descend from the construction and from the area bounds of [18]. □

We finally remark that there exists a drawing algorithm that computes 4-bend compatible drawings that are more readable than those computed by Algorithm THREE-BEND TREE. Although the area of these drawings is still $O((n+m)^2)$, they have optimal crossing angular resolution, i.e., edges cross only at right angles. Drawings of this type are called *RAC drawings* and are widely studied in the literature [13,14].

5 Discussion

We initiated the study of a new problem in graph drawing, i.e., computing a drawing Γ of a non-planar graph G such that a desired subgraph $S \subseteq G$ is crossing-free in Γ. In the setting where edges are straight-line segments and S is a spanning tree of G, we showed that Γ does not always exist; also, we provided existential and algorithmic results for meaningful subfamilies of spanning trees and we described a linear-time testing and drawing algorithm when S is a spanning triconnected subgraph. One of the main problems still open in this setting is the following: *Given a graph G and a spanning tree S of G, what is the complexity of deciding whether $\langle G, S\rangle$ admits a straight-line compatible drawing?* This problem can be also studied when S is a biconnected spanning subgraph, trying to extend the characterization of Theorem 4. Another interesting problem is to extend the results of Lemmas 1 and 2 in order to give a characterization of what spanning trees S of a complete graph can be always realized.

Allowing bends on the edges of $G \setminus S$, a drawing Γ exists for any given spanning tree S; we described several efficient algorithms that offer different compromises between drawing area and number of bends. Also, in this setting we have a characterization of which pairs $\langle G, S\rangle$ admit a 1-bend compatible drawing when S is a biconnected spanning subgraph. Namely, a necessary and sufficient condition is that S has a planar embedding such that for each edge e of $G \setminus S$ the end-vertices of e belong to the same face of S (as for Condition 1 of Theorem 4). Given such an embedding one can: (i) add a dummy vertex inside each face of S and connect it to all the vertices of the face; (ii) construct a planar straight-line drawing of the resulting graph, and (iii) construct a 1-bend compatible drawing where each edge (u, v) of $G \setminus S$ has a bend-point coinciding with the dummy vertex of the face containing u and v. A small perturbation of the bend-points will avoid that two of them coincide. An algorithm that tests the condition above can be derived as a simplification of the algorithm in [3], used to test the existence of a Simultaneous Embedding with Fixed Edges (SEFE) of two graphs [6]. Finally, we remark that Algorithm ONE-BEND TREE can be adapted to find a 1-bend compatible drawing when S is an outerplanar graph with the same bounds stated by Theorem 6. Many problems for k-compatible drawings are still open. Among them: trying to reduce the area bounds when S is a tree and devising algorithms for computing grid 1-bend compatible drawings of feasible $\langle G, S\rangle$ when S is biconnected.

References

1. Ackerman, E.: On the maximum number of edges in topological graphs with no four pairwise crossing edges. Discrete & Computational Geometry 41(3), 365–375 (2009)
2. Ackerman, E., Tardos, G.: On the maximum number of edges in quasi-planar graphs. Journal of Combinatorial Theory, Ser. A 114(3), 563–571 (2007)
3. Angelini, P., Di Battista, G., Frati, F., Patrignani, M., Rutter, I.: Testing the simultaneous embeddability of two graphs whose intersection is a biconnected or a connected graph. Journal of Discrete Algorithms 14, 150–172 (2012)
4. Angelini, P., Binucci, C., Da Lozzo, G., Didimo, W., Grilli, L., Montecchiani, F., Patrignani, M., Tollis, I.G.: Drawings of non-planar graphs with crossing-free subgraphs. ArXiv e-prints 1308.6706 (September 2013)
5. Bárány, I., Rote, G.: Strictly convex drawings of planar graphs. Documenta. Math. 11, 369–391 (2006)
6. Blasiüs, T., Kobourov, S.G., Rutter, I.: Simultaneous embedding of planar graphs. In: Tamassia, R. (ed.) Handbook of Graph Drawing and Visualization. CRC Press (2013)
7. Brandenburg, F.J., Eppstein, D., Gleißner, A., Goodrich, M.T., Hanauer, K., Reislhuber, J.: On the density of maximal 1-planar graphs. In: Didimo, W., Patrignani, M. (eds.) GD 2012. LNCS, vol. 7704, pp. 327–338. Springer, Heidelberg (2013)

8. Buchheim, C., Chimani, M., Gutwenger, C., Jünger, M., Mutzel, P.: Crossings and planarization. In: Tamassia, R. (ed.) Handbook of Graph Drawing and Visualization. CRC Press (2013)
9. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing. Prentice Hall, Upper Saddle River (1999)
10. Di Giacomo, E., Didimo, W., Liotta, G., Montecchiani, F.: h-quasi planar drawings of bounded treewidth graphs in linear area. In: Golumbic, M.C., Stern, M., Levy, A., Morgenstern, G. (eds.) WG 2012. LNCS, vol. 7551, pp. 91–102. Springer, Heidelberg (2012)
11. Di Giacomo, E., Didimo, W., Liotta, G., Montecchiani, F.: Area requirement of graph drawings with few crossings per edge. Computational Geometry 46(8), 909–916 (2013)
12. Didimo, W.: Density of straight-line 1-planar graph drawings. Information Processing Letters 113(7), 236–240 (2013)
13. Didimo, W., Eades, P., Liotta, G.: Drawing graphs with right angle crossings. Theoretical Computer Science 412(39), 5156–5166 (2011)
14. Didimo, W., Liotta, G.: The crossing angle resolution in graph drawing. In: Pach, J. (ed.) Thirty Essays on Geometric Graph Theory. Springer (2013)
15. Eades, P., Hong, S.H., Katoh, N., Liotta, G., Schweitzer, P., Suzuki, Y.: Testing maximal 1-planarity of graphs with a rotation system in linear time - (extended abstract). In: Didimo, W., Patrignani, M. (eds.) GD 2012. LNCS, vol. 7704, pp. 339–345. Springer, Heidelberg (2013)
16. Hong, S.-H., Eades, P., Liotta, G., Poon, S.-H.: Fáry's theorem for 1-planar graphs. In: Gudmundsson, J., Mestre, J., Viglas, T. (eds.) COCOON 2012. LNCS, vol. 7434, pp. 335–346. Springer, Heidelberg (2012)
17. Jansen, K., Woeginger, G.J.: The complexity of detecting crossingfree configurations in the plane. BIT Numerical Mathematics 33(4), 580–595 (1993)
18. Kant, G.: Drawing planar graphs using the canonical ordering. Algorithmica 16(1), 4–32 (1996)
19. Knauer, C., Schramm, É., Spillner, A., Wolff, A.: Configurations with few crossings in topological graphs. Computational Geometry 37(2), 104–114 (2007)
20. Korzhik, V.P., Mohar, B.: Minimal obstructions for 1-immersions and hardness of 1-planarity testing. Journal of Graph Theory 72(1), 30–71 (2013)
21. Kowalik, L., Kurowski, M.: Short path queries in planar graphs in constant time. In: Larmore, L.L., Goemans, M.X. (eds.) STOC 2003, pp. 143–148. ACM (2003)
22. Kratochvìl, J., Lubiv, A., Nešetřil, J.: Noncrossing subgraphs in topological layouts. SIAM Journal on Discrete Mathematics 4(2), 223–244 (1991)
23. Pach, J., Shahrokhi, F., Szegedy, M.: Applications of the crossing number. Algorithmica 16(1), 111–117 (1996)
24. Pach, J., Tóth, G.: Graphs drawn with few crossings per edge. Combinatorica 17(3), 427–439 (1997)
25. Rivera-Campo, E., Urrutia-Galicia, V.: A sufficient condition for the existence of plane spanning trees on geometric graphs. Computational Geometry 46(1), 1–6 (2013)
26. Suk, A.: k-quasi-planar graphs. In: Speckmann, B. (ed.) GD 2011. LNCS, vol. 7034, pp. 266–277. Springer, Heidelberg (2011)
27. Tutte, W.T.: How to draw a graph. Proceedings of the London Mathematical Society s3-13(1), 743–767 (1963)
28. Valtr, P.: On geometric graphs with no k pairwise parallel edges. Discrete & Computational Geometry 19(3), 461–469 (1998)