Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 26, no. 1, pp. 171–198 (2022)
DOI: 10.7155/jgaa.00587

Upward planar drawings with two slopes

Jonathan Klawitter[1] *Tamara Mchedlidze*[2]

[1]University of Würzburg, Germany
[2]Utrecht University, The Netherlands

Submitted: November 2021	Reviewed: March 2022	Revised: April 2022
Accepted: May 2022	Final: May 2022	Published: June 2022
Article type: Regular paper		Communicated by: G. Liotta

Abstract. In an upward planar 2-slope drawing of a digraph, edges are drawn as straight-line segments in the upward direction without crossings using only two different slopes. We investigate whether a given upward planar digraph admits such a drawing and, if so, how to construct it. For the fixed embedding scenario, we give a simple characterisation and a linear-time construction by adopting algorithms from orthogonal drawings. For the variable embedding scenario, we describe a linear-time algorithm for single-source digraphs, a quartic-time algorithm for series-parallel digraphs, and a fixed-parameter tractable algorithm for general digraphs. For the latter two classes, we make use of SPQR-trees and the notion of upward spirality. As an application of this drawing style, we show how to draw an upward planar phylogenetic network with two slopes such that all leaves lie on a horizontal line.

1 Introduction

When we visualize directed graphs (digraphs for short) that model hierarchical relations with node-link diagrams, we traditionally turn edge directions into geometric directions by letting each edge point upward. Aiming for visual clarity, we would like such an upward drawing to be planar, that is, no two edges should cross [14]. If this is possible, the resulting drawing is called *upward planar drawing*; see Figure 1 (a). Interestingly, as Di Battista and Tamassia [15] have shown, every upward planar drawing can be turned into one where each edge is drawn with a single line segment; such a *straight-line drawing* may however require an exponentially large drawing area [17].

Another important class of drawings are *(planar) orthogonal drawings*, where edges are drawn as sequences of horizontal and vertical line segments [14]; see Figure 1 (b). This drawing style is commonly used for schematic drawings such as VLSI circuit design and UML diagrams. Schematic drawings that allow more than two slopes for edge segments include hexalinear and octilinear drawings, which find application in metro maps [41]. In general, the use of only few geometric

E-mail addresses: jo.klawitter@gmail.com (Jonathan Klawitter) t.mtsentlintze@uu.nl (Tamara Mchedlidze)

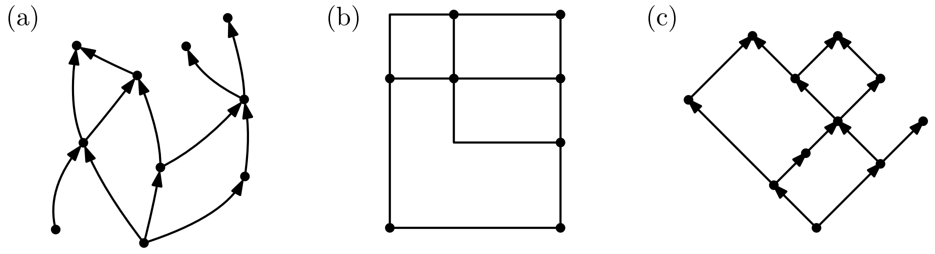

Figure 1: (a) An upward planar drawing; (b) an orthogonal drawing; (c) an upward planar 2-slope drawing.

primitives (such as different slopes) in a graph drawing facilitates a low visual complexity; a common quality measure for drawings [47].

In recent years, the interest in upward planar drawings that use only few different slopes has grown. For example, among other results, Bekos et al. [4] showed that every so-called bitonic *st*-graph with maximum degree Δ admits an upward planar drawing where every edge has at most one bend and the edges segments use only Δ distinct slopes. Di Giacomo et al. [19] provided complementary results by proving that also every series-parallel digraph admits a 1-bend upward planar drawing on Δ distinct slopes; their drawings also have optimal angular resolution. Brückner et al. [9] considered level-planar drawings, that is, upward planar drawings where each vertex is drawn on a predefined integer y-coordinate (its level), with a fixed slope set. In this paper, we continue this recent trend. In particular, we study bendless upward planar drawings that use only two different slopes. An example of such a drawing is shown in Figure 1 (c). Some of our results can be extended to 1-bend upward planar drawings. We now define these drawing concepts more precisely and list related work.

Upward planarity. An *upward planar drawing* of a directed graph G is a planar drawing of G where every edge (u, v) (i.e., an edge directed from u to v) is drawn as a monotonic upward curve from u to v. A digraph is called *upward planar* if it admits an upward planar drawing. Two upward planar drawings of the same digraph are *topologically equivalent* if the left-to-right orderings of the incoming and outgoing edges around each vertex coincide in the two drawings. An *upward planar embedding* is an equivalence class of upward planar drawings. An upward planar digraph is called *upward plane* if it is equipped with an upward planar embedding.

A necessary though not sufficient condition for upward planarity is acyclicity [5]. Moreover, Garg and Tamassia [24] showed that testing upward planarity is NP-complete for general digraphs. While the digraphs used in their reduction contain vertices with in- and outdegree higher than two, such vertices can be split into mulitple vertices of maximum in- and outdegree at most two without losing any of the properties required in their proofs. It is thus also NP-complete to test upward planarity for digraphs with in- and outdegree at most two. On the positive side, there exist several fixed-parameter tractable algorithms for general digraphs [10, 20, 26] and polynomial time algorithms for single source digraphs [6], series-parallel digraphs [20], outerplanar digraphs [43], and triconnected digraphs [5]. Moreover, upward planarity can be decided in polynomial time if the embedding is specified [5].

ℓ-bend k-slope drawings. In an *ℓ-bend drawing* of a graph G each edge is drawn with at most $\ell + 1$ line segments; equivalently, each edge has at most ℓ bends. An *ℓ-bend k-slope drawing* of G is an ℓ-bend drawing of G where every edge segment has one of at most k distinct slopes. From now on and if not further specified, we refer to bendless (or 0-bend) k-slope drawings simply as *k-slope drawings*.

Note that orthogonal drawings are 2-slope drawings without a bound on the number of bends. Tamassia [49] showed that a planar orthogonal drawing with minimum total number of bends of a plane graph on n vertices can be computed in $\mathcal{O}(n^2 \log n)$ time. Rhaman et al. [45] gave necessary and sufficient conditions for a subcubic plane graph to admit a bendless orthogonal drawing. For drawings of cubic graphs in 3D, Eppstein [23] considered bendless orthogonal graph drawings where two vertices are adjacent if and only if two of their coordinates are equal.

Given a graph G, the minimum number k of slopes needed for G to admit a k-slope drawing is called the *slope number* of G [53]. In the planar setting, this is the *planar slope number* of G. Both these numbers have been studied extensively. For example, Pach and Pálvölgyi [42] showed that the slope number of graphs with maximum degree 5 can be arbitrarily large. Further results, include bounds on slope numbers of graph classes such as trees, 2-trees, planar 3-trees, outerplanar graphs [21,33,38,39], cubic graphs [40], and subcubic graphs [18,34]. Determining the planar slope number is hard in the existential theory of the reals [28].

Upward planar 2-slope drawings. The focus of this paper lies on (bendless) upward planar 2-slope drawings. We consider only the slope set $\{-\pi/4, \pi/4\}$ and denote it by $\{\nwarrow, \nearrow\}$, since an upward planar 2-slope drawing on any two slopes can be morphed into an upward planar 2-slope drawing with the slopes $\nwarrow$ and $\nearrow$ – imagine this as (un)skewing a partial grid; see Figure 2. Note that a natural lower bound on the upward planar slope number of a graph is given by its maximum in- and outdegree. Hence, we assume that the graphs considered in this paper have maximum in- and outdegree at most two.

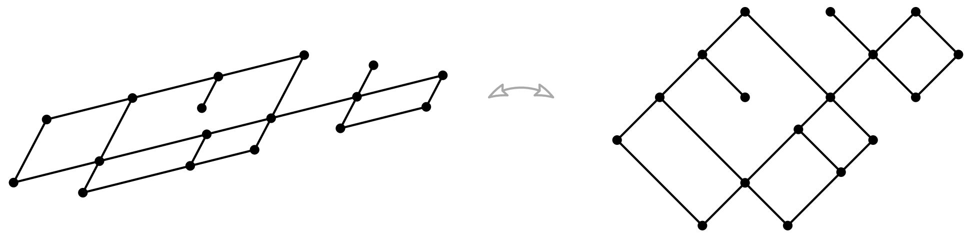

Figure 2: Two upward planar 2-slope drawings of the same graph on different slope sets – edge directions are given implicitly. Using an affine transformation one can transform a drawing on any size-two slope set into one on $\{\nwarrow, \nearrow\}$.

Bachmaier et al. [1], Brunner and Matzeder [8], and Bachmaier and Matzeder [3] studied straight-line drawings of ordered and unordered rooted trees on orthogonal grids with k directions for $k \in \{4, 6, 8\}$. Some of their drawing styles are also upward planar. A classical result of Crescenzi et al. [11] shows that any binary tree with n vertices admits an upward planar 2-slope drawing in $\mathcal{O}(n \log n)$ area. Concerning more complex graphs, upward planar drawings with few slopes for lattices have been studied by Czyzowicz et al. [13] and Czyzowicz [12]. As mentioned above, Bekos et al. [4] and Di Giacomo et al. [19] considered such drawings for st-graph and series-parallel graphs but also allowed bends. In a companion paper to the current one, Klawitter and Zink [36] study upward planar k-slope drawings for $k \geq 3$ and among other results show that it is NP-hard to decide whether an outerplanar digraph admits an upward planar 3-slope drawing.

Phylogenetic networks. Our interest in upward planar 2-slope drawings also stems from the problem of visualizing phylogenetic networks. *Phylogenetic trees* and *networks* are used to model the evolutionary history of a set of taxa like species, genes, or languages [22, 31, 48]. The precise definition of phylogenetic networks and their drawing conventions may vary widely depending on the particular use case. For instance, vertices may have timestamps that should be represented in the drawings or leaves may be required to be placed on the same height. In combinatorial phylogenetics the following definition is commonly used [31]: A *phylogenetic network* is a rooted digraph where the leaves are labelled bijectively with a set of taxa. Inner vertices are either *tree vertices* that have indegree one and outdegree two or *reticulations* that have indegree two and outdegree one; see Figure 3 (a). A network without reticulations is a *phylogenetic tree*; see Figure 3 (b,c).

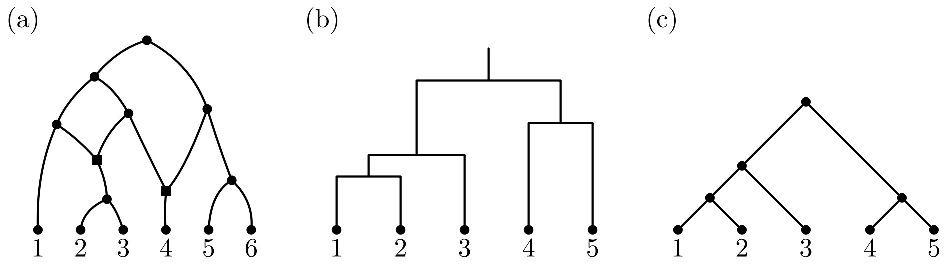

Figure 3: (a) A phylogenetic network with two reticulations; (b) a phylogenetic tree drawn as rectangular cladogram and (c) upward planar with two slopes.

There exist different drawing styles for phylogenetic trees such as rectangular or circular cladograms [2, 29]. If the focus is on the topology of the tree (and thus the taxonomy), a common drawing style is upward planar with 2-slope and all leaves aligned on a line. Theoretical work to adapt classical drawing styles from phylogenetic trees to phylogenetic networks has been carried out by Huson et al. [29,31,37]. A different approach has been taken by Tollis and Kakoulis [51], who propose a drawing style similar to treemaps for a special class of phylogenetic networks. There also exist several software tools to draw phylogenetic networks [7, 30, 32, 46, 52]. Here we are interested in drawing upward planar phylogenetic networks with two slopes and the additional constraint that all leaves lie on a horizontal line; see Figure 3 (c).

Contribution. In this paper we investigate the following decision problem. Given a digraph G of in- and outdegree at most two, decide whether it admits an upward planar 2-slope drawing. We distinguish the fixed and variable embedding scenario, that is, whether G is already equipped with an upward planar embedding or not. In the former case, we give a simple characterisation of when a drawing exists; see Section 3. By making use of orthogonal drawing algorithms, we also show how to construct a drawing (if it exists) in linear time. In addition, if no upward planar 2-slope drawing exists, we describe how to obtain an upward planar 1-bend 2-slope drawing of G with minimum number of bends.

For the variable embedding scenario, we check whether graphs of different graph classes admit an upward planar 2-slope drawing, based on the results of Section 3. In Section 4.1, we show that for a single-source digraph G, checking whether an upward planar 2-slope drawing of G exists can be done starting from any single upward planar embedding of G. In the affirmative, a suitable upward planar embedding can be derived and a drawing constructed in linear time. For series-parallel digraphs (Section 4.3) and general digraphs (Sections 4.4 and 4.5), we derive a quartic-time and a fixed-parameter tractable algorithm, respectively. These algorithms are based on Didimo et al.'s algorithms for upward planarity testing [20].

Lastly, we show how to compute 2-slope drawings of upward planar phylogenetic networks, where all leaves lie on a horizontal line in linear time; see Section 5. We conclude with a short discussion and open problems.

2 Preliminaries

Let G be an upward plane digraph with maximum in- and outdegree two. We assume, without loss of generality, that G is connected and let n denote the number of vertices of G or the graph currently under consideration. We use (u, v) to denote an edge of G that is directed from u to v. For two vertices $u, v \in V(G)$, we say that u precedes v if there is a directed path from u to v.

If a vertex v of G has two incoming edges, then based on the left-to-right ordering of the edges around v, it is natural to talk about the *left* and the *right* incoming edge of v. If an edge e is the only incoming edge at v, we call e the *sole* incoming edge of v. The same holds for outgoing edges; see Figure 4 (a). We say that a vertex v has face f to the *left (right)* if f is the face left (resp. right) of v's leftmost (resp. rightmost) incoming and leftmost (resp. rightmost) outgoing edge (if they exist).

A *2-slope assignment* ϕ is a mapping from the edges of G to the slopes $\nearrow$ and $\nwarrow$, that is, $\phi : E(G) \to \{\nwarrow, \nearrow\}$. We say ϕ is a *consistent 2-slope assignment* if

- every left (right) incoming edge of a vertex is assigned the slope $\nearrow$ (resp. $\nwarrow$), and
- every left (right) outgoing edge of a vertex is assigned the slope $\nwarrow$ (resp. $\nearrow$).

An edge (u, v) that is the sole outgoing edge of u and the sole incoming edge of v may have either slope. A digraph G together with a consistent 2-slope assignment ϕ forms an *upward planar 2-slope representation* $U_G = (G, \phi)$. To avoid cumbersome notation, we simply write 2-slope representation.

Suppose G contains an edge $e = (u, v)$ that is the left outgoing edge of u and left incoming edge of v. Let f be the face to the right of e; see Figure 4 (b). We then call e a *bad edge* with respect to f since e obstructs a consistent 2-slope assignment. Note, however, that e is not a bad edge with respect to the face f' left of e; see again Figure 4 (b). The same holds with "left" and "right" in reversed roles.

Let U_G be a 2-slope representation of G. Let $a = (e_1, v, e_2)$ be a triplet such that v is a vertex of the boundary of a face f and e_1, e_2 are incident edges of v that are consecutive on the boundary of f in counterclockwise direction. The following definitions follow Bertolazzi et al. [6] and Didimo et al. [20] though are adjusted to also encapsulate geometric properties induced by U_G. The triplet a is called an *angle* of f. We can categorise angles into three groups, namely, a is

- a *large angle* if e_1 and e_2 span a 270° angle in f,
- a *small angle* if e_1 and e_2 span a 90° angle in f, or
- a *flat angle* if e_1 and e_2 span a 180° angle in f

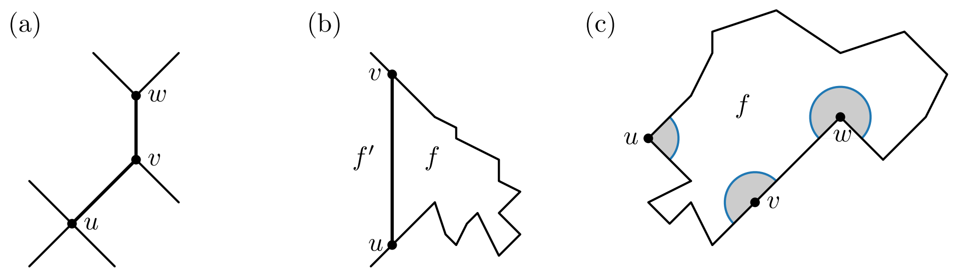

Figure 4: (a) The edge (u, v) is the right outgoing edge of u and left incoming edge of v, the edge (v, w) is the sole outgoing edge of v; (b) the edge (u, v) is a bad edge with respect to f; (c) u is a left-switch spanning a small angle, v spans a flat angle and is thus not a switch, and w is a sink-switch spanning a large angle.

with respect to the slopes assigned to e_1 and e_2. A vertex v of G is called a *local source* with respect to a face f if v has two outgoing edges on the boundary of f. A *local sink* is defined analogously. Furthermore, we call v a *switch* with respect to f if the slopes of e_1 and e_2 differ; for example, every local source is a switch. We further categorise switches by the angle they span and where they lie on the boundary of f; see Figure 4 (c). A switch v is a *large switch* if e_1 and e_2 span a large angle at f and a *small switch* otherwise; note that there can be no "flat" switches. We call v a *source-switch* or *sink-switch* if v is a local sink or local source, respectively. Otherwise, if e_1 and e_2 have f to the right (left), then v is a *left-switch* (resp. *right-switch*). Note that an inner face f of G contains exactly four small switches more than large switches and that the outer face contains four large switches more than small switches. An inner (the outer) face f is *rectangular* if it contains exactly four small (resp. large) switches.

Assume for now that G is biconnected. The following definitions are illustrated in Figure 5. A *split pair* $\{u, v\}$ of G is either a separation pair or a pair of adjacent vertices. A *split component* of G with respect to the split pair $\{u, v\}$ is either an edge (u, v) (or (v, u)) or a maximal subgraph G' of G such that G' contains u and v and $\{u, v\}$ is not a split pair of G'. Let G' be such a split component with respect to the split pair $\{u, v\}$. If G' is equipped with an upward planar embedding, then we define the *flip* of G' as the change of the embedding of G' by reversing the edge ordering of every vertex of G'.

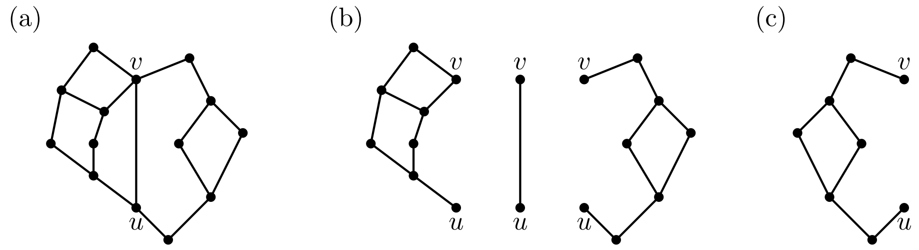

Figure 5: (a) A biconnected digraph with split pair $\{u, v\}$, (b) which induces three split components; (c) flip of the third split component.

3 Fixed embedding

In this section we consider the problem of whether an upward plane digraph admits an upward planar 2-slope drawing under the given fixed embedding. As noted above, a bad edge obstructs the existence of a consistent 2-slope assignment and thus of a 2-slope representation for G. We show that the absence of any bad edges is not only necessary but also sufficient.

Since upward planar 2-slope drawings are related to orthogonal drawings, we can make use of techniques used to construct them. The classical algorithm by Tamassia [49], which constructs an orthogonal drawing of a plane graph with minimum number of bends, works in three steps; refer also to Di Battista et al. [14, Chapter 5]. It starts with a plane graph and constructs a so-called *orthogonal representation*, a description of the shapes of the faces. The second step, called *refinement*, subdivides each face into rectangles. Finally, the third step performs a so-called *compaction* – it assigns coordinates to the vertices with the goal to minimize the area of the drawing. The technique for constructing an orthogonal representation cannot be directly applied for the construction of an upward planar 2-slope drawing, as it does not preserve the upwardness of edges. However, assuming a 2-slope representation is already given, we can adopt the refinement algorithm by Tamassia [49] and the compaction algorithm by Di Battista et al. [14] for our purposes. In the following lemma we describe a modified version of Tamassia's algorithm that refines the faces of a 2-slope representation; we explain how to obtain a 2-slope representation in Theorem 1. For this, recall that a switch is a triplet (e_1, v, e_2) consisting of a vertex and its two incident edges along a face in counterclockwise order where e_1 and e_2 have different slopes.

Lemma 1 *Let G be an upward plane digraph on n vertices with a 2-slope representation $U_G = (G, \phi)$. Then, in $\mathcal{O}(n)$ time, G can be refined into a digraph $\bar{G}$ that contains only rectangular faces and such that G is a topological minor of $\bar{G}$ respecting ϕ.*

Proof: If every face of G is already rectangular, then we are done and $\bar{G} = G$. Assume that this is not the case and let f be a non-rectangular inner face of G. We describe how to refine f into rectangular faces.

First, traverse the boundary of f counterclockwise and store in each switch pointers to its preceding and subsequent switch. Next, starting at any switch, traverse the circular sequence of switches in counterclockwise order. Let u be the first encountered large switch that is preceding a small switch v. Note that such u exists since f is not rectangular but contains at least four small switches. Without loss of generality, assume that u is a sink-switch. (The cases when u is a source-, left-, or right-switch work analogously.) Let w be the subsequent switch of v. If w is a large switch, then add a vertex x and the edges (u, x) and (w, x) with slopes $\nearrow$ and $\nwarrow$, respectively; see Figure 6 (a). Otherwise, if w is a small switch (and thus a right-switch), then subdivide the outgoing edge of w with a new vertex x, and add the edge (u, x) with slope $\nearrow$. Assign the slope $\nwarrow$ to the two edges resulting from the subdivision. This ensures that ϕ is respected by G as topological minor of $\bar{G}$; see Figure 6 (b). In either of the two cases, the result is a rectangular face f_1 and a face f_2 with one less small and one less large switch than f. Let f_2 now take the role of f and store the preceding switch of u and the subsequent switch of w as preceding and subsequent switches of x, respectively. If x is a small switch, continue the traversal with the switch preceding x instead of with x. Therefore, if a large switch precedes u in f, the process directly continues with a refinement step without having to potentially traverse the whole circular sequence first. Stop when only four switches are left and when f is thus rectangular. Note that this process runs in linear time in terms of the size of f and that it only adds as many new vertices as the number of large switches that f contains.

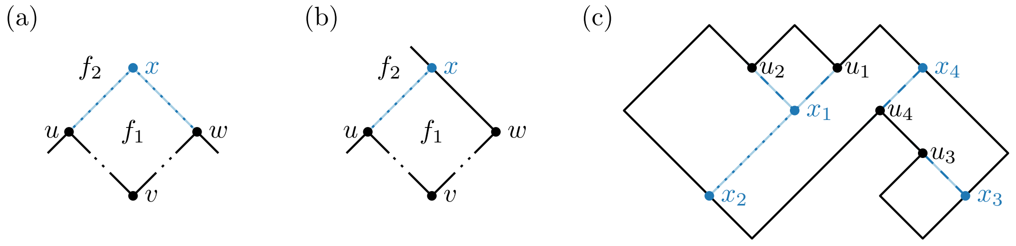

Figure 6: How to refine non-rectangular faces of G to obtain $\bar{G}$ when a large switch is followed (a) by a small and a large switch or (b) by two small switches. On the right, the large switch u_4 precedes the large switch u_3 and is thus processed after u_3.

Repeat this procedure for every non-rectangular inner face of G. If the outer face f_0 of G is non-rectangular, apply the analogous procedure with the difference that the goal is to remove all small switches such that f_0 only contains four large switches. Further note that here a large switch u preceding a small switch v exists since f_0 being non-rectangular implies that there is at least one small switch.

Let $\bar{G}$ be the resulting digraph where all faces are rectangular. By construction, $\bar{G}$ has G as topological minor and size in $\mathcal{O}(n)$. Furthermore, since the boundary of every face of G was traversed only twice, this refinement algorithm runs in $\mathcal{O}(n)$ time. □

We can now prove the main theorem of this section.

Theorem 1 *Let G be an upward plane digraph with n vertices. Then the following statements are equivalent.*

(F1) G admits an upward planar 2-slope drawing.

(F2) G admits a 2-slope representation U_G.

(F3) G contains no bad edge.

Moreover, there exists an $\mathcal{O}(n)$-time algorithm that tests if G satisfies (F3) and constructs an upward planar 2-slope drawing of G in the affirmative case.

Proof: Note that (F1) implies (F2) and (F2) implies (F3). We first show that (F3) implies (F2) and then how to construct a drawing (F1) from (F2).

Whether G contains a bad edge can easily be checked in $\mathcal{O}(n)$ time. Suppose it does not and thus satisfies (F3). Construct a consistent 2-slope assignment for G as follows. Go through all edges of G (in any order). For an edge e, if it is a left (right) incoming edge, assign to it slope $\nearrow$ (resp. $\nwarrow$). Otherwise, if it is a left (right) outgoing edge, assign to it slope $\nwarrow$ (resp. $\nearrow$). We claim that since e is not a bad edge, there is no conflict. Assume otherwise, namely, that $e = (u, v)$ gets, say, slope $\nwarrow$ from u and slope $\nearrow$ from v. However, then u must be the left outgoing edge at u and the left incoming edge at v, which makes e a bad edge thus contradiction our assumption. If e is both a sole incoming and a sole outgoing edge, assign it an arbitrary slope, say $\nwarrow$. Together with the already given upward planar embedding of G, this slope assignment yields a 2-slope representation U_G of G.

Next, we construct an upward planar 2-slope drawing of G. Use Lemma 1 to obtain a 2-slope representation $U_{\bar{G}}$ of an upward planar digraph $\bar{G}$ in which every face is rectangular in $\mathcal{O}(n)$ time.

Note that rotating $\bar{G}$ clockwise by 45°, makes the slope $\nwarrow$ vertical and the slope $\nearrow$ horizontal and we get an orthogonal representation of a graph where every face is rectangular. Therefore we can apply the linear-time compaction algorithm by Di Battista et al. [14, Theorem 5.3]. This algorithm assigns edge lengths and computes coordinates while handling the vertical and orthogonal direction of a orthogonal representation independently. Hence, applying the algorithm to $U_{\bar{G}}$, the edges with slopes $\nearrow$ and $\nwarrow$ are handled independently and keep their slopes. (Note that a, say, vertical edge (u, v) with length c in the orthogonal drawing corresponds to v being c units above and to the left of u in an upward planar 2-slope drawing.) As a result, we get an upward planar 2-slope drawing of $\bar{G}$, which we can reduce to a drawing of G. Since the three steps run in $\mathcal{O}(n)$ time each, the claim on the running time follows. □

Suppose we have an upward plane digraph G with k bad edges. Since G admits no upward planar 2-slope drawing, it is natural to ask whether G admits an upward planar 1-bend 2-slope drawing. In particular, if such a drawing exist, is it enough to bend only the k bad edges? Using Theorem 1 we can answer this question affirmatively.

Corollary 2 *Let G be an upward plane digraph with n vertices, maximum in- and outdegree at most two, and with k bad edges. Then G admits an upward planar 1-bend 2-slope drawing with k bends. Moreover, without changing the embedding, this is the minimum number of bends that can be achieved.*

Proof: Subdivide every bad edge once to obtain a graph G'. Then apply Theorem 1 to obtain an upward planar 2-slope drawing of G'. Since a bad edge e of G is neither a sole incoming nor a sole outgoing edge, the two edges obtained from e in G' have different slopes. Hence, by turning every subdivision vertex into a bend we get an upward planar 1-bend 2-slope drawing of G.

Since even a single bad edge obstructs a 2-slope representation and since bending a non-bad edge clearly does not eliminate any other bad edges either, it follows that k bends are also necessary. □

Note that G may admit an upward planar 1-bend 2-slope drawing with less or no bends if the embedding is changed.

4 Variable embedding

In this section we consider the problem of whether a given upward planar digraph G of a particular graph class admits an upward planar 2-slope drawing under any upward planar embedding. We start with two general observations, before we consider the class of single-source digraphs, where upward planarity can be tested in linear time, and then continue with the more complex classes of series-parallel digraphs and general digraphs.

Let G be an upward planar digraph with maximum in- and outdegree at most two. Suppose G contains a leaf ℓ. Note that removing ℓ from G does not change whether G admits an upward planar 2-slope drawing. Moreover, we may reduce G to a digraph without leaves, obtain a drawing of the reduced digraph (if possible), and then add the leaves to obtain a 2-slope drawing of G. Removing and later restoring leaves takes only linear time.

While leaves are no obstruction, transitive edges are. However, note that not all bad edges have to be transitive edges; see Figure 8.

Observation 3 *A transitive edge of an upward planar digraph G is a bad edge in any upward planar embedding of G.*

Proof: Let $e = (u, v)$ be a transitive edge of G. By definition, there is a directed path P from u to v different from e. Since P may not cross e, it enters v from the same side (left or right) as it leaves u in any upward planar embedding of G and hence e is always a bad edge. □

4.1 Single-source digraphs

For a single-source digraph G, our idea is to first compute an arbitrary upward planar embedding of G with the linear-time algorithm of Bertolazzi et al. [6]. We then check whether there are any bad edges and, if so, whether they can be fixed with small changes to the embedding. To this end, we need the following lemmata.

Lemma 2 *An upward planar single-source digraph G contains no bad edge with respect to the outer face.*

Proof: Consider an upward planar embedding of G and suppose $e = (u, v)$ is a bad edge with respect to the outer face f_0; see Figure 7 (a). Let e' be the second incoming edge of v. Then v is a local sink of f_0 such that if e and e' would be drawn with slopes $\nearrow$ and $\nwarrow$ accordingly, then f_0 would have a small angle at v. However, this implies that there are two local sources (w_1 and w_2 in Figure 7 (a)) of f_0 that are also sources of G. This is a contradiction to G being a single-source digraph. □

Lemma 3 *Let G be an upward planar single-source digraph with maximum in- and outdegree at most two. Then in any upward planar embedding of G, there are at most two bad edges with respect to the same face.*

Proof: Let G be an upward plane single-source digraph and let f be a face of G. Note that a bad edge e with respect to f is incident to a local sink v of a face f where v lies between its two incoming edges in a counterclockwise traverse of the boundary of f. In other words, in an upward planar drawing of G, f would have a (small) angle of less than 180° at v. If there are three or more bad edges with respect to f, then f contains at least two such local sinks (like v_1 and v_2 in Figure 7 (b)). There is then at least one local source u for f that would span a large angle in f, i.e., more than 180°. However, u is also a source of G and since it is not the source for the outer face, it is not the only source of G. This is a contradiction to G being a single-source digraph. Figure 7 (c) gives an example of a face with two bad edges. □

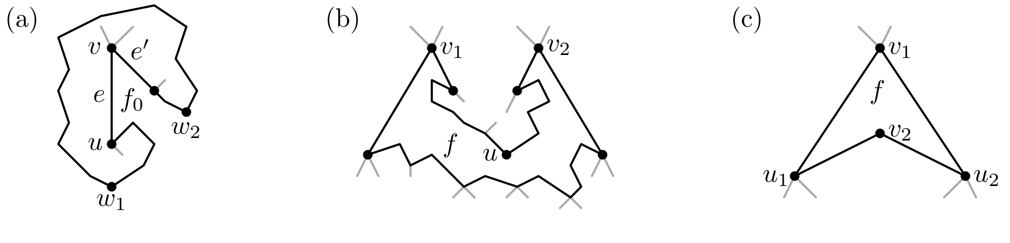

Figure 7: (a) A bad edge with respect to the outer face f_0 implies the existence of at least two sources w_1 and w_2; (b) a face f that has four bad edges of which two are adjacent to the local sink v_1 of f and two are adjacent to the local sink v_2 of f. However, f also contains a source u; (c) a face f with two bad edges (u_1, v_1) and (u_2, v_1) that can be a face of a single source digraph.

Lemma 4 *Let G be an upward plane single-source digraph with maximum in- and outdegree at most two and with bad edges $\{e_1, e_2, \ldots, e_k\}$. Let e_1 be a bad edge with respect to face f. Deciding whether G admits an upward planar embedding with bad edges $\{e_2, \ldots, e_k\}$ can be done in $\mathcal{O}(|f|)$ time.*

Proof: Let $e_1 = e = (u, v)$, let e_u be the second outgoing edge of u, and let e_v be the second incoming edge of v. Note that e_v and e_u are also on the boundary of f and that by Lemma 2, f is not the outer face. We claim that e cannot be a bridge. Assume otherwise and let G_u and G_v be the components of $G \setminus \{e\}$ that contain u and v, respectively. Note that both G_u and G_v have a source each and v cannot be the source of G_v. This implies that G has two sources, which is a contradiction to G being a single-source digraph. Let f' be the second face with e on its boundary, which exists since e is not a bridge. Without loss of generality, assume that f' is to the left and f to the right of e.

For G to admit an upward planar embedding where e is not a bad edge, it must be possible to change, without loss of generality, the order of e_u and e at u, that is, e_u and e need to become the left and right outgoing edges of u, respectively. If e_u is a bridge (and thus has f as left and right face), then we can swap e and e_u at u and flip the component that does not contain u of $G \setminus \{e_u\}$; see Figure 8 (a). We can find out whether e_u is a bridge with a single traverse of f. Furthermore, clearly, this flip does not introduce a new bad edge.

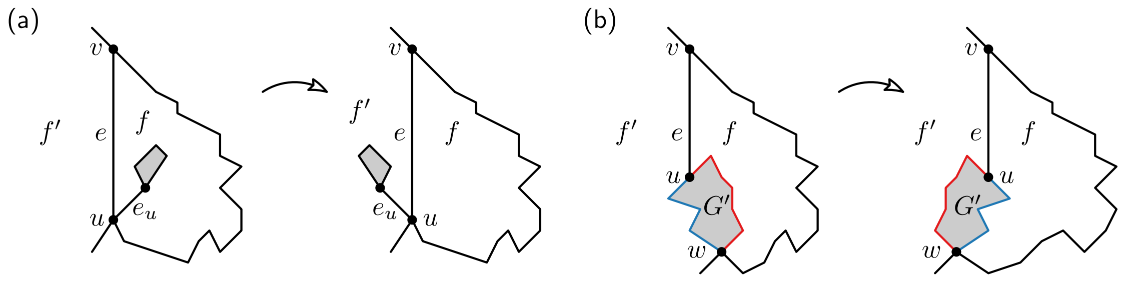

Figure 8: Two scenarios where a bad edge e with respect to f in a single-source digraph can be repaired: (a) e_u is a bridge and can be flipped from f to f'; (b) the digraph G' between f and f' and between the split pair $\{w, u\}$ can be flipped.

Otherwise, if e_u is not a bridge, observe that we need to flip a subgraph G' of G that contains e_u and that is enclosed by f' and f; see Figure 8 (b). For such G' to exist, there must be a vertex w that forms a split pair with u, and that has f to the right and f' to the left. It can be seen as a split pair $\{u, w'\}$ without this property does not yield a flippable digraph G' (as in Figure 9 (a)) and if no such a split pair exists at all, then the triconnectedness implies that e_v, e, and e_u always lie on the boundary of the same face (as in Figure 9 (b)). Assuming now that such w exists, we can define G' as the digraph consisting of all split components of G with respect to $\{w, u\}$ that do not contain v. To repair e, we flip G' as shown in Figure 8 (b), that is, we reverse the order of incoming and outgoing edges for each vertex in G' except for w, where we only reverse the order of the outgoing edges. Note that the flip cannot introduce a new bad edge since w is not a local sink or local source of f or f'. Furthermore, note that such w precedes u, since otherwise both G' and $G \setminus G'$ would contain a source (that is not w) each, which contradicts G being a single-source digraph; see Figure 9 (c). Hence, we may find w with a traverse of f.

Lastly, we show that changing the order of e and e_v at v is possible only when the case above applies where the order of e and e_u at u can be changed. To begin with, note that e_v cannot be a

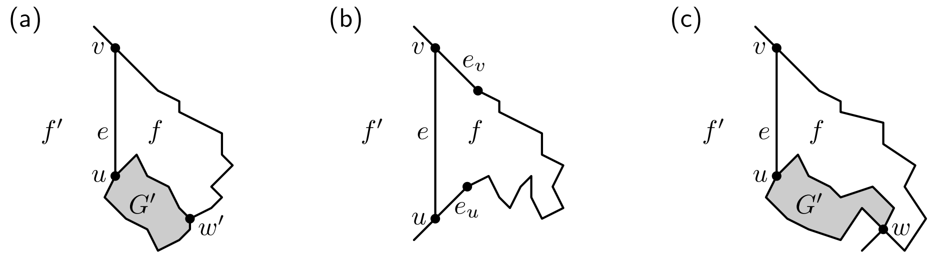

Figure 9: Scenarios where a bad edge e with respect to f in a single-source digraph cannot be repaired (a) since G' cannot be flipped or (b) since there is no split pair $\{w, u\}$; (c) here G' can be flipped, but the digraph is not a a single-source digraph.

bridge, since G is a single-source digraph. Hence, we would need to flip again a digraph G' that contains e_v and that is enclosed by f and f'. Such G' would imply a split pair $\{w', v\}$, where, however, w' would also serve as split pair $\{w', u\}$ as in Figure 8 (b).

Since we can check whether e_u is a bridge or find a suitable w with a single traverse of f, the claim on the running time holds. □

From Lemma 4, we get that an upward plane single-source digraph admits an upward planar 2-slope drawing (possibly for a different upward planar embedding) if every bad edge can be fixed. We may thus check every bad edge and perform the necessary flips. Since digraphs to flip may be nested, executing simply one flip after the other could be costly. We now show how to keep the running time linear.

Theorem 4 *Let G be an upward planar single-source digraph with n vertices. An upward planar 2-slope drawing of G can be computed in $\mathcal{O}(n)$ time, if one exists.*

Proof: As noted above, we may assume that G does not contain a leaf. A linear-time algorithm to compute an upward planar 2-slope drawing of G then works as follows. First, compute an upward planar embedding of G with the algorithm of Bertolazzi et al. [6] in $\mathcal{O}(n)$ time. Second, to identify all bad edges in $\mathcal{O}(n)$ time it suffices to traverse the boundary of each face since every bad edge is bad with respect to exactly one face. Third, for each bad edge $e = (u, v)$ with respect to a face f, check whether it can be repaired with Lemma 4. To keep track of where we have to flip the edge order at vertices, we use two types of markers. A green marker at a vertex x indicates that the outgoing edges of x should be swapped and that both incoming and outgoing edges of the vertices "above" x should be reversed. We thus mark u green if e can be repaired because the respective edge e_u is a bridge and we mark w green if e can be repaired because of a respective split pair $\{w, u\}$. Furthermore, we mark e red to tells us to stop reversing edge orders. Both the check and the marking can be done in $\mathcal{O}(|f|)$ time. Since by Lemma 3 any face contains at most two bad edges, this step takes overall also only $\mathcal{O}(n)$ time.

Suppose every bad edge can be fixed. Then run a BFS on G that starts at the source. During the traversal, remember along each path the number of green marked vertices minus the number of encountered red edges. Then for each vertex v, use the parity of this number to decide whether its edge orders have to be reversed; if odd, then reverse, and if even, then do not. Vertices marked green need to be handled appropriately. This takes again only linear time and as a result we get an upward planar embedding of G that admits a 2-slope representation U_G. Finally, apply the algorithm from Theorem 1 on U_G to compute an upward planar 2-slope drawing of G. □

Note that in Lemma 4, whether a bad edge is bad in any upward planar embedding of G is independent from whether another edge is bad in any upward planar embedding of G. Hence, we can also use Theorem 4 and Corollary 2 to minimise the number of bends in a 1-bend 2-slope drawing of G.

Corollary 5 *Let G be an upward planar single-source digraph with n vertices and maximum in- and outdegree two. An upward planar 1-bend 2-slope drawing of G with the minimum number of bends can be computed in $\mathcal{O}(n)$ time.*

4.2 SPQR-trees and upward spirality

Didimo et al. [20] described algorithms to compute upward planar embeddings of biconnected series-parallel and general digraphs. They then use a result from Healy and Lynch [27] about combining biconnected blocks of upward planar digraphs to get rid of the biconnectivity condition. We follow their approach closely. In particular, we also use the notions of SPQR-trees and upward spirality (with the latter tailored to our needs) on which Didimo et al.'s approach heavily relies on and tailor them to our needs. We refer to Didimo et al. [20] for the precise definition of SPQR-trees (for undirected graphs and then derived for digraphs), though recall the main concepts in this section.

Let G be a biconnected digraph and let $e = (s, t)$ be any edge of G called *reference edge*. An *SPRQ-tree* T of G with respect to e represents a decomposition of G with respect to its triconnected components [16, 25]. As such, it also represents all planar embeddings of G with e on the outer face. Starting with the split pair $\{s, t\}$, the decomposition is constructed recursively on the split pairs of G. More precisely, T is a rooted tree where every node is of type S, P, Q, or R: Q-nodes represent single edges, S-nodes and P-nodes represent *series components* and *parallel components*, and R-nodes represent triconnected (*rigid*) components; see Figure 10 for an example. Each node μ in T has associated a biconnected multigraph $skel(\mu)$, called the *skeleton* of μ, in which the children of μ and its reference edge are represented by a virtual edge. The root of T is a Q-node representing (s, t). The child of μ is now defined recursively as follows:

Trivial case. If G consists of exactly two parallel edges between s and t, then μ is a Q-node representing the edge (s, t) parallel to the reference edge and the skeleton $skel(\mu)$ is G.

Parallel case. If the split pair $\{s, t\}$ has three or four split components $G_1, G_2, \ldots, G_k$ (more are not possible with our degree restrictions), then μ is a P-node. The skeleton $skel(\mu)$ consists of k parallel edges between s and t that represent the reference edge G_1 and the components G_i, $2 \leq i \leq k$.

Series case. If $\{s, t\}$ induces two split components $e = (s, t)$ and $\bar{G}$, then μ is an S-node. If $\bar{G}$ is a chain of biconnected components $G_1, \ldots, G_k$ with cut vertices $c_1, \ldots, c_{k-1}$ $(k \geq 2)$, then $skel(\mu)$ is the cycle t, e, s, e_1, c_1, $\ldots$, c_{k-1}, e_k, t where e_i represents G_i.

Rigid case. Otherwise μ is an R-node. Let $\{s_1, t_1\}$, $\ldots$, $\{s_k, t_k\}$ be the maximal split pairs of G with respect to $\{s, t\}$. Let G_i be the union of split components of $\{s_i, t_i\}$ except the one containing e. Then $skel(\mu)$ is obtained from G by replacing each subgraph G_i with e_i.

The skeleton of the root consists of two parallel edges, e and a virtual edge representing the rest of the graph. Each node μ of T that is not a Q-node has children $\mu_1, \ldots, \mu_k$, in this order such that μ_i is a child of μ based on $G_i \cup e_i$ with respect to e_i. The *pertinent graph* G_μ for a node μ of

T represents the full subgraph of G in the SPQR-tree rooted at μ. The end vertices of $e_i = (s_i, t_i)$ are called the *poles* of μ, of *skel*(μ), and of G_μ. Refer to Figure 10 for an example and note that every edge is represented by a Q-node. We assume that T is in its *canonical form*, that is, every S-node of T has exactly two children. The canonical form can be derived from a non-canonical form in linear time [20].

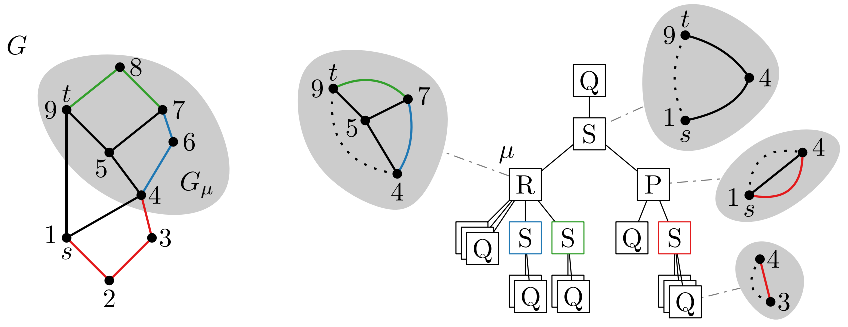

Figure 10: A digraph G and its (non-canonical) SPQR-tree with respect to the reference edge $\{s, t\}$. The skeletons of some nodes are depicted with the virtual edge drawn dashed. The pertinent graph G_μ of the R-node μ is highlighted in G.

Assume that G is equipped with an st-numbering (based on its underlying undirected graph and with respect to the reference edge $\{s, t\}$). Let μ be a node of T with poles u and v such that u precedes v in the st-numbering. Then u is the *first pole* and v is the *second pole* of μ.

Assume that a pertinent graph G_μ of a node μ of T admits an upward planar 2-slope drawing and let U_{G_μ} be a 2-slope representation of G_μ. Let u and v be the poles of μ and let $w \in \{u, v\}$. The *pole category* t_w of the pole w is the way the edges that lie in the outer face of G_μ are incident to w and which slopes they got assigned under U_{G_μ}. Note that edges incident to w that lie in the interior of G_μ do not affect the pole category of w. The sixteen possible pole categories of split components are shown in Figure 11. Two poles w and w' with pole category t_w and t'_w, respectively, are *compatible* if t_w and t'_w can be combined into a pole category of higher degree. For example, combining iR and iL gives iRiL.

Next we define upward spirality, which measures how much a split component is "rolled up". The pertinent graph G_μ of a node μ of T may have 2-slope representations with different spirality. Let U_{G_μ} be a 2-slope representation of G_μ with first pole u and second pole v. Let P be an undirected path in U_{G_μ}. Two subsequent edges $\{x, y\}$ and $\{y, z\}$ of P define a *right (left) turn* if P makes a 90° clockwise (resp. counterclockwise) turn at y according to U_{G_μ}. Define the *turn number* $\mathrm{n}(P)$ of P as the number of right turns minus the number of left turns of P. Let P_l and P_r be the clockwise and counterclockwise paths from u to v along the outer face, respectively. We define the *upward 2-slope spirality* σ of U_{G_μ} as $\sigma(U_{G_\mu}) = \frac{\mathrm{n}(P_l)+\mathrm{n}(P_r)}{2}$. See Figure 12 for examples. Note that the turn number of a path is bounded by its length. Therefore, the number of possible values for the upward 2-slope spirality of U_{G_μ} is in $\mathcal{O}(n)$ (and in $\mathcal{O}(d)$ where d is the diameter of U_{G_μ}). For technical reasons that become apparent later, when we store the upward 2-slope spirality of a 2-slope representation, we also store the values $\mathrm{n}(P_l)$ and $\mathrm{n}(P_r)$.

Let μ be a node of T with first pole u and second pole v. A *feasible tuple* of μ is a tuple $\tau_\mu = \langle U_{G_\mu}, \sigma, t_u, t_v\rangle$ where U_{G_μ} is an upward 2-slope representation of G_μ with pole categories t_u and t_v and upward 2-slope spirality σ. Two feasible tuples of μ are *spirality equivalent* if they have

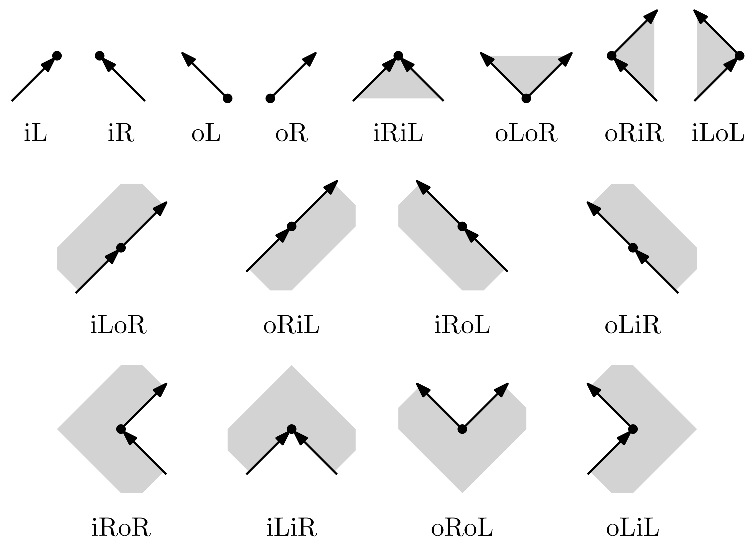

Figure 11: Pole categories in a 2-slope representation; gray areas indicate where the interior of the graph lies.

the same upward 2-slope spirality and pole categories. A *feasible set* $\mathcal{F}_\mu$ of μ is a set of all feasible tuples of μ such that there is exactly one representative tuple for each class of spirality equivalent tuples of μ.

4.3 Series-parallel digraphs

Let G be a biconnected series-parallel digraph, that is, an SPQR-tree (or rather SPQ-tree) of G contains no R nodes. Our goal is to test the existence of an upward planar embedding that does not contain a bad edge and thus the existence of a 2-slope representation of G. The idea of the algorithm is as follows. Pick a reference edge e and construct the respective SPQR-tree T of G. In a post-order traversal of T, compute the feasible set of each node. This is straightforward for

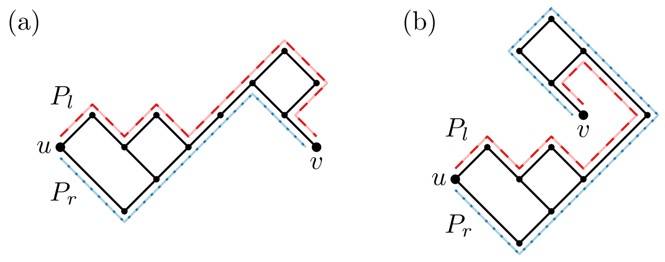

Figure 12: Two different 2-slope representations of the same digraph for different upward planar embeddings: (a) The paths P_l and P_r have turn number 1 and 0, and so the upward 2-slope spirality is 0.5; (b) the paths P_l and P_r have turn number −3 and −4, and so the upward 2-slope spirality is −3.5

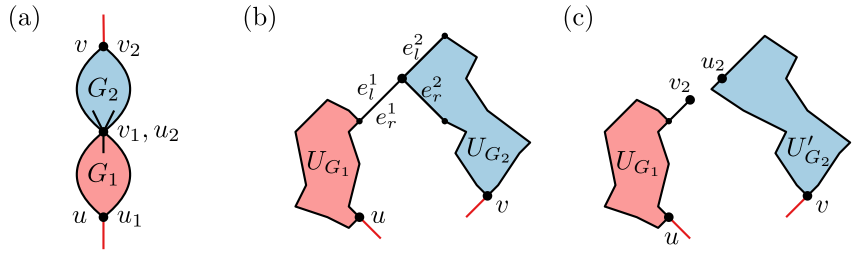

Figure 13: Illustration of a series composition of split components G_1 and G_2: (b) U_{G_1} and U_{G_2} are compatible and the upward 2-slope spirality can be computed based on $\sigma(U_{G_1})$, $\sigma(U_{G_2})$, and the turns at e_l^1 and e_l^2 and at e_r^1 and e_r^2; (c) U_{G_1} and U'_{G_2} are not compatible.

Q-nodes. For an S- or P-node μ, try to combine feasible tuples of the children of μ to feasible tuples of μ. The pole categories and upward 2-slope spirality values make it easier to check whether a composition admits an upward planar 2-slope representation. If this leads to a non-empty feasible set for the root of T, we can construct a drawing. Otherwise, we try again with another reference edge.

Let μ be a Q-node of T with first pole u and second pole v and suppose G_μ is the edge (u, v); the case where G_μ is (v, u) is handled analogously. Then there are only two upward 2-slope representation of G_μ, namely when (u, v) is assigned the slope $\nwarrow$ or the slope $\nearrow$. Therefore, $\mathcal{F}_\mu$ has size two. The following two lemmata show how to compute the feasible set of an S-node and a P-node of T.

Lemma 5 *Let G be a biconnected digraph with n vertices and T be an SPQR-tree of G. Let μ be an S-node of T with children μ_1 and μ_2. Given the feasible sets $\mathcal{F}_{\mu_1}$ and $\mathcal{F}_{\mu_2}$, the feasible set $\mathcal{F}_\mu$ can be computed in $\mathcal{O}(n^2)$ time.*

Proof: To compute a feasible tuple τ of $\mathcal{F}_\mu$ we check for all pairs $\tau_1 \in \mathcal{F}_{\mu_1}$ and $\tau_2 \in \mathcal{F}_{\mu_2}$ whether they can be combined. More precisely, let u be the first and v the second pole of μ; refer to Figure 13. Let u_i be the first and v_i be the second pole of μ_i, $i \in \{1, 2\}$. Without loss of generality, assume that $u = u_1$, $v_1 = u_2$, and $v_2 = v$. For each pair of feasible tuples $\tau_1 = \langle U_{G_{\mu_1}}, \sigma_1, t_{u_1}, t_{v_1}\rangle \in \mathcal{F}_{\mu_1}$ and $\tau_2 = \langle U_{G_{\mu_2}}, \sigma_2, t_{u_2}, t_{v_2}\rangle \in \mathcal{F}_{\mu_2}$ check whether t_{v_1} and t_{u_2} are compatible. In other words, check whether G_{μ_1} and G_{μ_2} can be plugged together at $v_1 = u_2$ under the slope assignments of $U_{G_{\mu_1}}$ and $U_{G_{\mu_2}}$ and with the reference edge on the outer face.

If the pole categories are compatible, the feasible tuple $\tau = \langle U_{G_\mu}, \sigma, t_u, t_v\rangle$ is given by $t_u = t_{u_1}$, $t_v = t_{v_2}$, U_{G_μ} as the series composition of $U_{G_{\mu_1}}$ and $U_{G_{\mu_2}}$ at the common vertex $u_2 = v_1$, and where σ can be computed as follows. For $i \in \{1, 2\}$, let e_l^i and e_r^i be the edges of $U_{G_{\mu_i}}$ that are incident to $u_2 = v_1$ and lie on the clockwise and counterclockwise path from u to v along the outer face, respectively; see Figure 13 (b). Note that e_l^i may coincide with e_r^i. For $j \in \{l, r\}$, let α_j be -1, 1, or 0 depending on whether e_j^1 and e_j^2 make a left, right, or no turn, respectively. The upward 2-slope spirality of U_{G_μ} is $\sigma = \sigma_1 + \sigma_2 + \frac{\alpha_l + \alpha_r}{2}$ (compare to Lemma 6.4 by Didimo et al. [20]). Store τ in $\mathcal{F}_\mu$ if $\mathcal{F}_\mu$ contains no feasible tuple with spirality equivalent to τ. Since $\mathcal{F}_{\mu_1}$ and $\mathcal{F}_{\mu_2}$ have $\mathcal{O}(n)$ tuples, $\mathcal{F}_\mu$ can be computed in $\mathcal{O}(n^2)$ time. □

Lemma 6 *Let G be a biconnected digraph with n vertices and T be an SPQR-tree of G. Let μ be a P-node of T with children $\mu_1, \ldots, \mu_k$ with $k \leq 4$. Given the feasible sets $\mathcal{F}_{\mu_1}, \ldots, \mathcal{F}_{\mu_k}$, the feasible set $\mathcal{F}_\mu$ can be computed in $\mathcal{O}(n)$ time.*

Proof: Let u be the first and v the second pole of μ (and its children). Since u and v have at most degree four in G, μ has at most four children (but at least two). We first consider the case where μ has three children. The cases where μ has two or four children are discussed at the end of this proof.

For $i \in \{1,2,3\}$, let G_i be the pertinent digraph of μ_i. Note that u and v have degree one in G_i. We can thus define e_i and e'_i as the edges of G_i incident to u and v, respectively. (If μ_i is a Q-node, then $e_i = e'_i$.) Let e_u and e_v be the edges of G incident to u and v, respectively, that lie outside of G_μ. (Note that $e_u = e_v$ is only possible in the final composition, where e_u is then also the reference edge.) We want to construct a 2-slope representation of G_μ such that the order of e_1, e_2, e_3, e_u at u is the reverse order of e'_1, e'_2, e'_3, e_v at v; see Figure 14 (a). Furthermore, the half edges representing e_u and e_v have to be on the outer face.

Suppose we pick a feasible tuple $\tau_1 = \langle U_{G_1}, \sigma_1, t_{u_1}, t_{v_1}\rangle \in \mathcal{F}_{\mu_1}$. Then t_{u_1} and t_{u_2} restrict the choices of pole categories of compatible upward 2-slope representations of G_2 and G_3. Likewise, σ_1 restricts these choices further; see Figure 14 (b) and (c). More precisely, we observe that the upward 2-slope spirality σ_2 and σ_3 of compatible U_{G_2} and U_{G_3} differ from σ_1 by at least two and at most six (compare to Lemma 6.6 by Didimo et al. [20]). Therefore, when trying all possible permutations of G_1, G_2, and G_3, we only have to consider $\mathcal{O}(1)$ feasible tuples in $\mathcal{F}_{\mu_2}$ and $\mathcal{F}_{\mu_3}$ instead of iterating over complete sets. Storing the feasible tuples in a hash table based on the spirality and pole categories we can find compatible τ_2 and τ_3 in constant time. Similar to the series-composition, we can compute the upward 2-slope spirality of the resulting upward 2-slope representation U_{G_μ} based on its categories and σ_i, $i \in \{1,2,3\}$. More precisely, for this purpose we not only stored each σ_i but also the respective values $\mathrm{n}(P_l)$ and $\mathrm{n}(P_r)$ to compute them. Therefore, we can take the appropriate values from the two 2-slope representations of U_{G_i}, $i \in \{1,2,3\}$, that are on the outer face. Overall, we get that by iterating once over $\mathcal{F}_{\mu_1}$ we can construct all feasible tuples of $\mathcal{F}_\mu$ in $\mathcal{O}(n)$ time.

The case where μ has four children is only possible in the final composition with the reference edge e. Here the algorithms works along the same line with the difference that the half edges e_u and e_v are replaced with e. The case where μ has two children, works analogously with the difference that for some feasible tuples of U_{G_1} there can now be more compatible upward spirality values for U_{G_2} than above. However, these are still $\mathcal{O}(1)$ many and the running time is thus not affected. $\square$

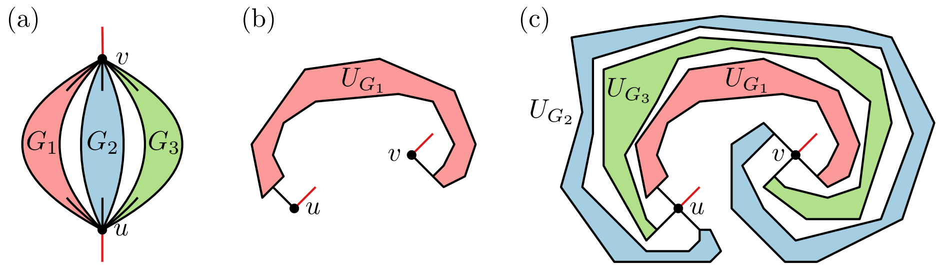

Figure 14: Illustration of a parallel compositions with three children and how picking an upward 2-slope representation of G_1 enforces the spirality of representation of G_2 and G_3.

Lastly, for the root composition we check in $\mathcal{O}(n)$ time whether the root of T, which is a Q-node representing e, can be combined with a feasible tuple of its child. In the affirmative case, we obtain an upward 2-slope representation of G. With all compositions described, we can now prove the main theorem of this section.

Theorem 6 *Let G be a biconnected series-parallel digraph with n vertices. There exists an $\mathcal{O}(n^4)$-time algorithm that tests if G admits an upward planar 2-slope drawing and, if so, that constructs such a drawing.*

Proof: Let e be an edge of G. Compute the (canonical) SPQR-tree T with respect to e of G, which can be done in $\mathcal{O}(n)$ time [20, 25]. In a post-order traversal of T, the algorithm computes the feasible set for every node of T. If the algorithm arrives at a node with an empty feasible set, its starts with another reference of G. Otherwise, the algorithm stops when it has constructed a feasible tuple for G. We can then use Theorem 1 to construct an upward planar 2-slope drawing. For one reference edge, this takes at most $\mathcal{O}(n^2)$ time per node by Lemmas 5 and 6. and since the size of T is linear in n, at most $\mathcal{O}(n^3)$ time in total. The total running time is thus in $\mathcal{O}(n^4)$. □

In Section 4.5 we explain how to handle non-biconnected series-parallel digraphs.

4.4 Biconnected digraphs

We extend the algorithm for biconnected series-parallel digraphs to general biconnected digraphs following again Didimo et al. [20]. The upward planarity check is again combined with finding a 2-slope representation. Let G be a biconnected digraph. Let T be the SPQR-tree of G with respect to a reference edge e. The algorithm computes again the feasible sets of the nodes of T in a post-order traversal. For Q-, S-, or P-nodes this works as before. Recall that to compute a feasible tuple of an S-node or P-node it suffices to look at the pole categories and upward spirality of its children. For R-node this connection is not as clear and we rely thus on a brute-force approach. More precisely, we compute the feasible set by considering all possible combinations of tuples for each virtual edge of $\mathit{skel}(\mu)$ to construct U_{G_μ}. If substitutions are successful, we have to check upward planarity and the existence of bad edges.

Lemma 7 *Let G be a biconnected digraph with n vertices and T be an SPQR-tree of G. Let μ be an R-node of T with children $\mu_1, \ldots, \mu_k$. Let d be the diameter of G_μ. Given the feasible sets $\mathcal{F}_{\mu_1}, \ldots, \mathcal{F}_{\mu_k}$, the feasible set $\mathcal{F}_\mu$ can be computed in $\mathcal{O}(d^k n^2)$ time.*

Proof: Note that since $\mathit{skel}(\mu)$ is triconnected, it has a unique planar embedding (up to mirroring), which we can compute in $\mathcal{O}(n)$ time. Note that, by Theorem 1, if $\mathit{skel}(\mu)$ contains a bad edge with respect to three non-virtual edges, then $\mathcal{F}_\mu$ is empty. Moreover, then G admits no upward planar 2-slope drawing and the main algorithm can stop. So suppose no such bad edge exists.

Construct a 2-slope representation of G_μ by substituting virtual edges with the respective 2-slope representations. More precisely, for all $i \in \{1, \ldots, k\}$, substitute e_i with the 2-slope representation $U_{G_{\mu_i}}$ of a feasible tuple in $\mathcal{F}_{\mu_i}$. Let U'_{G_μ} denote the partial upward 2-slope representation of G_μ during this process, that is, U'_{G_μ} consists of an embedding of G_μ and the 2-slope assignment for all edges of the $U_{G_{\mu_i}}$, for $i \in \{1, \ldots, k\}$. At each pole of a child μ_i in U'_{G_μ} check whether the 2-slope representations of the substituted parts are conflicting. If this check fails, backtrack and try another feasible tuple. Suppose that it is successful for all poles of all μ_i. Then test the upward planarity of U'_{G_μ} with the flow-based upward planarity algorithm for triconnected digraphs by Bertolazzi et al. [5] where the assignment of switches to faces is given for the substituted parts (derived from U'_{G_μ}). This flow-based algorithm runs in $\mathcal{O}(n^2)$ time.

If U'_{G_μ} is upward planar, check whether U'_{G_μ} contains any bad edge and, if not, extend U'_{G_μ} to a 2-slope representation U_{G_μ} of G_μ (compare to Lemma 8.2 by Didimo et al. [20]). Suppose there is an edge (x, y) on the outer face of U_{G_μ} that is the sole outgoing edge of x and the sole incoming edge of y. Note that the choice of slope for (x, y) does not influence the spirality of U_{G_μ}, since the angles formed at x and y always add up to the same value; see Figure 15. Lastly, compute the upward 2-slope spirality of U_{G_μ} in $\mathcal{O}(n)$ time to obtain a feasible tuple for $\mathcal{F}_\mu$. By backtracking and trying the remaining feasible tuples of the μ_i, we complete the computation of $\mathcal{F}_\mu$.

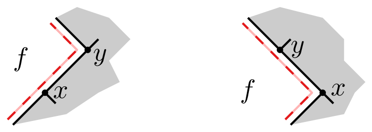

Figure 15: In an R-node μ, the slope choice for an edge (x, y) that is the sole outgoing edge at x and sole incoming edge at y along the outer face does not effect the upward spirality U_{G_μ}.

Note that d is an upper bound on the upward 2-slope spirality of a split component and thus each μ_i has $\mathcal{O}(d)$ feasible tuples. Therefore the algorithm tries at most $\mathcal{O}(d^k)$ combinations of feasible tuples of the k children of μ. Hence the feasible set of an R-node μ can be computed in $\mathcal{O}(d^k n^2)$ time, where the factor $\mathcal{O}(n^2)$ comes from the flow based upward planarity test. □

Let d denote the maximum diameter of a split component of G and let t denote the number of nontrivial triconnected components of G, i.e., series components, parallel components, and rigid components. Didimo et al. [20] have shown that in Lemma 7 we can also bound the running time with $\mathcal{O}(d^t n^2)$ and further, in total, compute the feasible sets of all R-nodes in $\mathcal{O}(d^t t n^2)$ time. Recall that the time needed to compute the feasible set of an S-node is bounded by the square of possible spirality values and is thus in $\mathcal{O}(d^2)$. Since we use the canonical form of SPQR-trees there are $\mathcal{O}(n)$ S-nodes. Therefore, we can compute the feasible sets of all S-nodes of T in $\mathcal{O}(d^2 n)$ time. Similarly, the feasible sets of all P-nodes of T can be computed in $\mathcal{O}(dt)$ time. Lastly, iterating over all SPQR-trees of G for the different choices of reference edges adds another factor of $\mathcal{O}(n)$. If a 2-slope representation of G has been found, we apply again Theorem 1 to compute a drawing. Hence, we get the following theorem.

Theorem 7 *Let G be a biconnected digraph with n vertices. Suppose that G has at most t non-trivial triconnected components, and that each split component has diameter at most d. Then there exists an $\mathcal{O}(d^t t n^3 + dtn + d^2 n^2)$-time algorithm that tests if G admits an upward planar 2-slope drawing and, if so, that constructs such a drawing of G.*

4.5 General digraphs

So far we have seen how to test whether a biconnected digraph admits a 2-slope representation. For a general digraph G, even if each of its biconnected components, called a *block*, has a 2-slope representation we may not be able to join the blocks; see Figure 16. In fact, even if all blocks of G being upward planar does not imply that G is upward planar [27]. Nonetheless, following Healy and Lynch [27], our strategy is to test for each block if its admits a 2-slope representation under some special conditions and, in the affirmative, join these representations.

When we want to merge the representations of two blocks, we have to take two things into consideration. Namely, we have to test whether their joined cut vertex c is on the outer face for

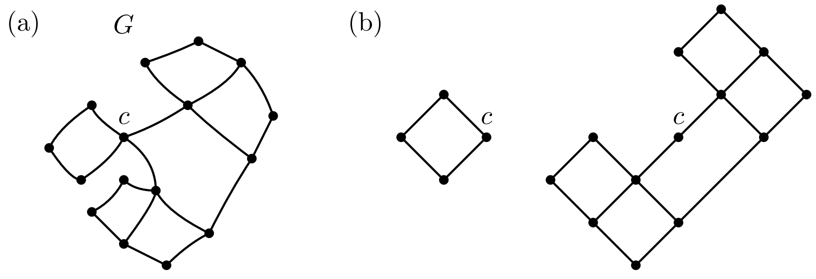

Figure 16: G does not admit a 2-slope representations (a), even though its blocks do (b).

one of the two blocks and whether their 2-slope representations fit together at c (just like poles and their pole categories). Before we get into detail on this, we recall what a block-cut tree is and explain how it gives a suitable order to process the blocks.

Block-cut tree. The *block-cut tree* $\mathcal{T}$ of G contains a vertex for each block and for each cut vertex, and an edge between a cut vertex c and each block that contains c. For G to admit a 2-slope representation, we must be able to root $\mathcal{T}$ at a block (with all edges oriented towards this root block) such that an edge $\{B, c\}$ between a block B and cut vertex c is oriented towards c only if B admits a 2-slope representation where c is on the outer face (see Figure 17 and compare to Lemma 3 by Chan [10]). Note that if we can root $\mathcal{T}$ at a block B', then any other block B has exactly one outgoing edge and B' has only incoming edges. For a block B with outgoing edge to a cut vertex c, we say B is a *block with respect to* c. Note that multiple blocks can be a block with respect to the same cut vertex.

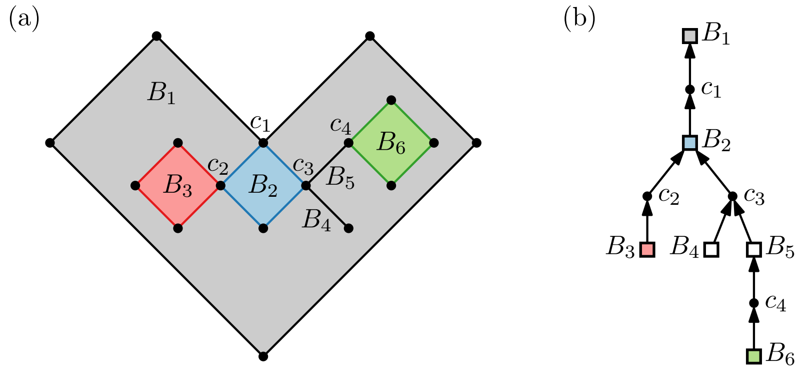

Figure 17: A digraph with six blocks and 2-slope representation (a) and the corresponding rooted block-cut tree (b).

Suppose we would know how to root $\mathcal{T}$. With a post-order traversal of $\mathcal{T}$, we could then try to find a 2-slope representation for each block B with respect to c of $\mathcal{T}$ that has c on the outer face. However, a priori we do not know at which block to root $\mathcal{T}$. Hence, our algorithm works as follows.

Algorithm. Given G, we can find its cut vertices and blocks and construct its block-cut tree $\mathcal{T}$ in $\mathcal{O}(n)$ time [50]. Since we do not know what block can function as root of $\mathcal{T}$ yet, we start at the leaves of $\mathcal{T}$ and work "inwards". Note that a leaf block B has only one edge $\{B, c\}$ in $\mathcal{T}$ to a cut vertex c (unless $B = G$) and we can thus provisionally direct $\{B, c\}$ to c. This yields a block with respect to a cut vertex – B with respect to c. Furthermore, during the algorithm, for at least one non-leaf block B either all or all but one of its neighboring blocks have been handled. We then direct each edge $\{B, c'\}$, where all other blocks adjacent to c' have been handled, towards B. If no undirected edge incident to B remains, then all neighboring blocks of B have been handled and B is the root. Otherwise, there remains exactly one undirected edge $\{B, c\}$. Therefore, during this ad hoc post-order traversal of $\mathcal{T}$, we can ensure that we always have at least one block B that is the root or for which we can provisionally direct the last undirected edge incident to B towards a cut vertex c in $\mathcal{T}$ i.e., B becomes a block with respect to c.

To process a block B with respect to c, we check whether B has a 2-slope representation U_B with (i) c on the outer face and (ii) additional constraints on the angles formed at all other cut vertices of B, which we describe in detail below. If this is the case, then we can finalize the direction of the edge $\{B, c\}$ towards c. Otherwise, if B does not admit such a 2-slope representation, then B has to be the root of $\mathcal{T}$. We then orient all edges of $\mathcal{T}$ towards B and continue in the remaining part of $\mathcal{T}$. If we later find that another block also needs to be the root of $\mathcal{T}$, then G does not admit a 2-slope representation. Furthermore, when we arrive back at B, we have to test whether B admits a 2-slope representation at all.

Note that for a block B with respect to c, where c has to be on the outer face of the 2-slope representation U_B, the algorithms from the previous two sections only have to consider an SPQR-tree with an edge incident to c as reference edge.

Angles of cut vertices. We now describe the additional constraints that have to be checked for a block B at all of B's cut vertices. More precisely, let B be a block with respect to c (or the root) and let c' be any other cut vertex of B if it has any. Depending on the degree of c (and c') in B and in the neighboring blocks of B, we have the following extra conditions on the angles at c and c' in U_B of B.

- Suppose c (or c') has degree one in B. Then B is a single edge, c is automatically on the outer face in any 2-slope representation of B, and the angle at c in U_B is insignificant.

- Suppose c has degree two in B and c has degree two in another block B_1; see Figure 18 (a). Then c has to have a large angle on the outer face in U_B, since otherwise it would not be able to attach to U_{B_1} such that B_1 is in the outer face of B. If the same case applies to c', then c' has to have a large angle in U_B, but not necessarily on the outer face.

- Suppose c has degree two in B and c has degree one in two other blocks B_1 and B_2. Then we first test if B admits a 2-slope representation where c has a large angle on the outer face; see Figure 18 (b). In this case, both B_1 and B_2 lie in the outer face of B. Otherwise, if c has indegree one and outdegree one in B, we test whether B admits a 2-slope representation where c forms a flat angle on the outer face; see Figure 18 (c). We test once such that B_1 lies on the outer face of B and once for B_2. In the former case, B_2 would lie in the interior of B and thus $\{B_2, c\}$ would be directed as (B_2, c); in the latter case, B_1 would lie in the interior of B and thus $\{B_1, c\}$ would be directed as (B_1, c). If neither is possible, then B has to be the root. For c' there are no restrictions under these conditions.

- The case where c (or c') has degree two in B and degree one in exactly one other block is similar to the previous case but simpler.
- Suppose c has degree three in B; see Figure 18 (d). Then c has to have a flat angle at c on the outer face. Otherwise B has to be the root. There are again no restrictions for c' under these conditions.

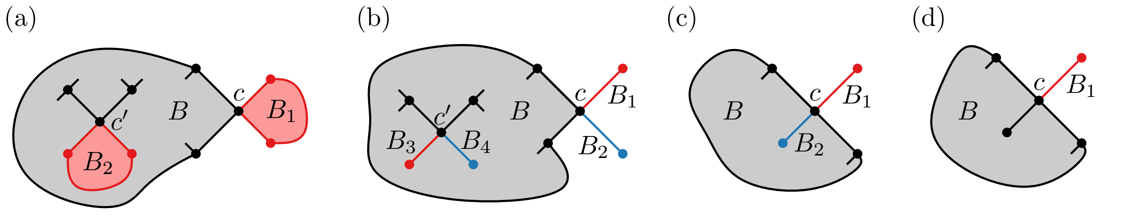

Figure 18: Conditions on the angles at c and c' in U_B for block a B with respect to c.

These conditions are clearly necessary and, following Healy and Lynch [27], also sufficient. Furthermore, they can easily be tested by the algorithms from the previous sections, where we (as observed above) can use an edge e incident to c as reference edge. More precisely, if c has degree two in B, then its two incident edges are merged in the root composition. Hence, at this step we only allow a merge with the desired angle at c, that is, we check if there there is 2-slope representation of the child node of e with suitable upward spirality. Otherwise, if c has degree three in B and has to form a flat angle, we take the sole outgoing or sole incoming edge of c in B as the reference edge for the SPQR-tree. In the root composition we then only allow a merge when there is the desired flat angle at c on the outer face.

Result. The total running time for our algorithm to test whether a general digraph admits a 2-slope representation and thus an upward planar 2-slope drawing is given by (i) a linear amount for the computation of $\mathcal{T}$, (ii) the sum of the checks for each block, and (iii) a linear amount for merging. Because each cut vertex lies in at most four blocks, the running time for (ii) is at most as much as if we tested a biconnected graph of the same size as G once. Hence, we get the following results.

Theorem 8 *Let G be a series-parallel digraph with n vertices. There exists an $\mathcal{O}(n^4)$-time algorithm that tests if G admits an upward planar 2-slope drawing and, if so, that constructs such a drawing.*

Theorem 9 *Let G be a digraph with n vertices. Let t be the maximum number of nontrivial triconnected components of a block of G, and d be the maximum diameter of a split component of a block of G. Then there exists an $\mathcal{O}(d^t tn^3 + dtn + d^2n^2)$-time algorithm that tests if G admits an upward planar 2-slope drawing and, if so, that constructs such a drawing of G.*

5 Phylogenetic networks

Recall from Section 1 that a phylogenetic network is a single-source digraph whose sinks are all leaves and whose non-sink, non-source vertices have degree three. In this section we show how to find an upward planar 2-slope drawing of a phylogenetic network N such that its leaves lie on a horizontal line – if N admits such a drawing. Since we want that all leaves are on the outer face, we first merge them into a single vertex and then apply the linear-time algorithm of Bertolazzi et al. [6] to test whether the resulting digraph N' is upward planar. Clearly, N' is upward planar if and only if N admits a desired upward planar embedding. In the affirmative case, let N now be an upward plane phylogenetic network such that its k leaves lie on the outer face. Further assume that N contains no bad edge or, in this case equivalently, no transitive edge (unlike the network in Figure 20 (a)).

In Section 3, we constructed an upward planar 2-slope drawing by implementing the refinement step, which augments all faces to rectangular faces, and by applying a compaction algorithm [35]. In order to obtain a drawing where all leaves lie on the same horizontal line, we apply this algorithm to the following augmentation $\bar{N}$ of N. Let $l_1, l_2, \dots, l_k$ be the leaves of N in clockwise order around the outer face. Add new vertices $v_1, v_2, \dots, v_{k-1}$ and edges $e_i = (l_i, v_i)$ and $e'_i = (l_{i+1}, v_i)$, $i \in \{1, \dots, k-1\}$; see Figure 19 (a). Then apply Theorem 1 to $\bar{N}$ to obtain an upward planar 2-slope drawing of $\bar{N}$ in $\mathcal{O}(n)$ time.

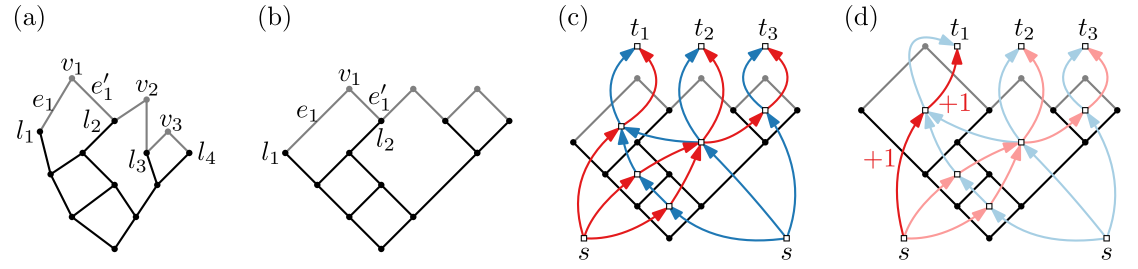

Figure 19: (a) The augmentation $\bar{N}$ of an upward planar phylogenetic network N; (b) an upward planar 2-slope drawing where l_1 and l_2 have different y-coordinates; (c) the dual flows G_l (red) and G_r (blue); (d) Propagating the length difference of e_1 and e'_1 through G_l.

We observe from Figure 19 (b) that two vertices l_i and l_{i+1} of $\bar{N}$ (neighboring leaves of N) have different y-coordinates if and only if e_i and e'_i have different lengths. This can be fixed by propagating these length differences through the drawing in the following way (Figure 19 (c–d)). Let G be the dual graph of $\bar{N}$. Furthermore, define G_l and G_r as the two subgraphs of G with $V(G) = V(G_l) = V(G_r)$ and where $E(G_l)$ and $E(G_r)$ are the dual edges of primal edges with slope $\nwarrow$ and $\nearrow$, respectively. In other words, G_l is the dual graph of $\bar{N}$ restricted to edges with slope $\nwarrow$. Direct every edge e^* in $E(G_l)$ ($E(G_r)$) with primal edge e from the left (resp. right) face of e to the right (resp. left) face of e. Assign to each dual edge flow equal to the length of its primal edge. Split the vertex corresponding to the outer face of $\bar{N}$ into a source s and $k-1$ sinks $t_1, t_2, \dots, t_{k-1}$ such that t_i has as incoming edges the dual edges of the primal edges e_i and e'_i; see Figure 19 (c). These dual graphs can be constructed in linear time.

Next, to adjust the heights of the leaves of N, for every pair e_i and e'_i, $i \in \{1, \dots, k-1\}$, if, say, e'_i is shorter than e_i, propagate the difference as flow backwards towards s through G_l; see Figure 19 (d). With one DFS on G_l and G_r each, all leaves can be handled simultaneously and in linear time. Lastly, since some edge lengths have been changed, update the coordinates of all vertices in $\bar{N}$ and remove the vertices v_i, $i \in \{1, \dots, k\}$, to obtain an upward planar 2-slope drawing of N. The following theorem summarises this section.

Theorem 10 *Let N be a phylogenetic network with n vertices and no transitive edge. If N admits an upward planar drawing with all its leaves on the outer face, then N admits and upward planar 2-slope drawing such that all its leaves lie on a horizontal line. Moreover, such a drawing can be constructed in $\mathcal{O}(n)$-time.*

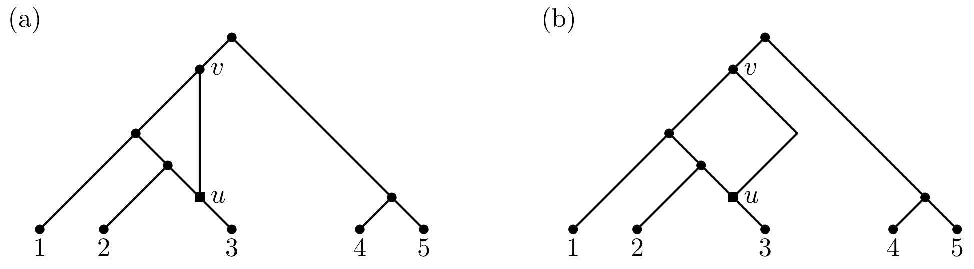

Figure 20: (a) A phylogenetic network with a transitive edge (u, v) does not admit an upward planar 2-slope drawing, (b) however, it admits an upward planar 1-bend 2-slope drawing.

Note that by Corollary 5 a phylogenetic network with m transitive edges admits a upward planar 1-bend 2-slope drawing with at most m bends; see Figure 20.

6 Concluding remarks

When considering the number of slopes in a graph drawing, one typically asks how many different slopes are necessary for a graph of certain graph class. Here we instead constrained the number of slopes to two and asked what digraphs can then be drawn upward planar. Our digraphs are thus limited to those that contain no transitive edges and have a small maximum degree. Beyond that, the difficulty of the problem depends on whether or not an upward planar embedding is given and on the complexity of the digraph.

We have shown that if the embedding is fixed then the question can be answered and, in the affirmative, a drawing constructed in linear time. In this case the problem boils down to whether there is a bad edge for the given embedding and, if not, to adapt algorithms for orthogonal drawings. However, even if there are bad edges present, allowing each of them to bend once is enough to obtain an upward planar 1-bend 2-slope drawing with the minimum number of bends. We conjecture that it is NP-hard to minimize the drawing area of an upward planar 2-slope drawing just like it is for orthogonal drawings [44]. It would be interesting to see a proof for this and how compaction algorithms for orthogonal drawings can be applied to upward planar drawings.

If a given digraph is not embedded yet, we first have to check whether the digraph is upward planar. For single-source digraph, we have seen that it suffices to find one upward planar embedding, which may then be altered to one without bad edges if it exists. For series-parallel and general digraphs we reused an approach by Didimo et al. [20] based on SPQR-trees and upward spirality to find a quartic time and a fixed-parameter tractable algorithm, respectively. An important difference is that our algorithm does not only compute upward planar embeddings for nodes of the SPQR-tree but also 2-slope representations. Through the degree restrictions the algorithm became simpler and can thus also consider other properties. It would be interesting to see whether the algorithm that computes an upward planar embedding of a single-source digraph can be modified to directly compute a 2-slope representation.

This research was motivated by drawings of phylogenetic networks. While we here assumed that a given phylogenetic network is upward planar, this is not a biologically motivated property of phylogenetic networks. One may argue that phylogenetic networks often have few reticulations (vertices with indegree two or higher), but even just two reticulations suffice to obstruct upward planarity. Hence, it would be interesting to have algorithms that can also draw non-upward planar phylogenetic network with two slopes.

The biggest challenges remain for drawings with more than two slopes. Our feeling is that while the complexity of developing algorithms to draw graphs with two slopes is manageable, three or more slopes increase the geometric interdependence dramatically. While the companion paper by Klawitter and Zink [36] started to investigate this, we would be happy to see more results on upward planar slope numbers of graphs.

Acknowledgements

We thank the reviewers for their helpful comments and suggestions.

References

[1] C. Bachmaier, F. J. Brandenburg, W. Brunner, A. Hofmeier, M. Matzeder, and T. Unfried. Tree drawings on the hexagonal grid. In I. G. Tollis and M. Patrignani, editors, *Graph Drawing*, pages 372–383. Springer, 2009. doi:10.1007/978-3-642-00219-9_36.

[2] C. Bachmaier, U. Brandes, and B. Schlieper. Drawing phylogenetic trees. In X. Deng and D.-Z. Du, editors, *Algorithms and Computation*, pages 1110–1121. Springer, 2005. doi:10.1007/11602613_110.

[3] C. Bachmaier and M. Matzeder. Drawing unordered trees on k-grids. *Journal of Graph Algorithms and Applications*, 17(2):103–128, 2013. doi:10.7155/jgaa.00287.

[4] M. A. Bekos, E. Di Giacomo, W. Didimo, G. Liotta, and F. Montecchiani. Universal Slope Sets for Upward Planar Drawings. In T. Biedl and A. Kerren, editors, *Graph Drawing and Network Visualization*, pages 77–91. Springer International Publishing, 2018. doi:10.1007/978-3-030-04414-5_6.

[5] P. Bertolazzi, G. Di Battista, G. Liotta, and C. Mannino. Upward drawings of triconnected digraphs. *Algorithmica*, 12(6):476–497, 1994. doi:10.1007/BF01188716.

[6] P. Bertolazzi, G. Di Battista, C. Mannino, and R. Tamassia. Optimal Upward Planarity Testing of Single-Source Digraphs. *SIAM Journal on Computing*, 27(1):132–169, 1998. doi:10.1137/S0097539794279626.

[7] A. Boc, A. B. Diallo, and V. Makarenkov. T-REX: a web server for inferring, validating and visualizing phylogenetic trees and networks. *Nucleic Acids Research*, 40(W1):W573–W579, 2012. doi:10.1093/nar/gks485.

[8] W. Brunner and M. Matzeder. Drawing ordered (k-1)-ary trees on k-grids. In U. Brandes and S. Cornelsen, editors, *Graph Drawing*, pages 105–116. Springer, 2011. doi:10.1007/978-3-642-18469-7_10.

[9] G. Brückner, N. D. Krisam, and T. Mchedlidze. Level-planar drawings with few slopes. In D. Archambault and C. D. Tóth, editors, *Graph Drawing and Network Visualization*, pages 559–572, 2019. doi:10.1007/978-3-030-35802-0_42.

[10] H. Chan. A Parameterized Algorithm for Upward Planarity Testing. In S. Albers and T. Radzik, editors, *Algorithms – ESA 2004*, pages 157–168. Springer, 2004. doi:10.1007/978-3-540-30140-0_16.

[11] P. Crescenzi, G. Di Battista, and A. Piperno. A note on optimal area algorithms for upward drawings of binary trees. *Computational Geometry*, 2(4):187–200, 1992. doi:10.1016/0925-7721(92)90021-J.

[12] J. Czyzowicz. Lattice diagrams with few slopes. *Journal of Combinatorial Theory, Series A*, 56(1):96–108, 1991. doi:10.1016/0097-3165(91)90025-C.

[13] J. Czyzowicz, A. Pelc, and I. Rival. Drawing orders with few slopes. *Discrete Mathematics*, 82(3):233–250, 1990. doi:10.1016/0012-365X(90)90201-R.

[14] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.

[15] G. Di Battista and R. Tamassia. Algorithms for plane representations of acyclic digraphs. *Theoretical Computer Science*, 61(2):175–198, 1988. doi:10.1016/0304-3975(88)90123-5.

[16] G. Di Battista and R. Tamassia. Incremental planarity testing. In *30th Annual Symposium on Foundations of Computer Science*, pages 436–441, 1989. doi:10.1109/SFCS.1989.63515.

[17] G. Di Battista, R. Tamassia, and I. G. Tollis. Area Requirement and Symmetry Display of Planar Upward Drawings. *Discrete & Computational Geometry*, 7:381–401, 1992. doi:10.1007/BF02187850.

[18] E. Di Giacomo, G. Liotta, and F. Montecchiani. Drawing subcubic planar graphs with four slopes and optimal angular resolution. *Theoretical Computer Science*, 714:51–73, 2018. doi:10.1016/j.tcs.2017.12.004.

[19] E. Di Giacomo, G. Liotta, and F. Montecchiani. 1-bend upward planar slope number of SP-digraphs. *Computational Geometry*, 90:101628, 2020. doi:10.1016/j.comgeo.2020.101628.

[20] W. Didimo, F. Giordano, and G. Liotta. Upward Spirality and Upward Planarity Testing. *SIAM Journal on Discrete Mathematics*, 23(4):1842–1899, 2010. doi:10.1137/070696854.

[21] V. Dujmović, D. Eppstein, M. Suderman, and D. R. Wood. Drawings of planar graphs with few slopes and segments. *Computational Geometry*, 38(3):194–212, 2007. doi:10.1016/j.comgeo.2006.09.002.

[22] M. Dunn. Language phylogenies. In C. Bowern and B. Evans, editors, *The Routledge Handbook of Historical Linguistics*, chapter 7. Routledge, 2014. doi:10.4324/9781315794013.ch7.

[23] D. Eppstein. The Complexity of Bendless Three-Dimensional Orthogonal Graph Drawing. *Journal of Graph Algorithms and Applications*, 17(1):35–55, 2013. doi:10.7155/jgaa.00283.

[24] A. Garg and R. Tamassia. On the Computational Complexity of Upward and Rectilinear Planarity Testing. *SIAM Journal on Computing*, 31(2):601–625, 2001. doi:10.1137/S0097539794277123.

[25] C. Gutwenger and P. Mutzel. A Linear Time Implementation of SPQR-Trees. In J. Marks, editor, *Graph Drawing*, pages 77–90. Springer, 2001. doi:10.1007/3-540-44541-2_8.

[26] P. Healy and K. Lynch. Two fixed-parameter tractable algorithms for testing upward planarity. *International Journal of Foundations of Computer Science*, 17(05):1095–1114, 2006. doi:10.1142/S0129054106004285.

[27] P. Healy and K. Lynch. Building blocks of upward planar digraphs. *Journal of Graph Algorithms and Applications*, 11(1):3–44, 2007. doi:10.7155/jgaa.00135.

[28] U. Hoffmann. On the Complexity of the Planar Slope Number Problem. *Journal of Graph Algorithms and Applications*, 21(2):183–193, 2017. doi:10.7155/jgaa.00411.

[29] D. H. Huson. Drawing Rooted Phylogenetic Networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 6(1):103–109, 2009. doi:10.1109/TCBB.2008.58.

[30] D. H. Huson and D. Bryant. Application of Phylogenetic Networks in Evolutionary Studies. *Molecular Biology and Evolution*, 23(2):254–267, 2005. doi:10.1093/molbev/msj030.

[31] D. H. Huson, R. Rupp, and C. Scornavacca. *Phylogenetic Networks: Concepts, Algorithms and Applications*. Cambridge University Press, 2010.

[32] D. H. Huson and C. Scornavacca. Dendroscope 3: An Interactive Tool for Rooted Phylogenetic Trees and Networks. *Systematic Biology*, 61(6):1061–1067, 2012. doi:10.1093/sysbio/sys062.

[33] V. Jelínek, E. Jelínková, J. Kratochvíl, B. Lidický, M. Tesař, and T. Vyskočil. The Planar Slope Number of Planar Partial 3-Trees of Bounded Degree. *Graphs and Combinatorics*, 29(4):981–1005, 2013. doi:10.1007/s00373-012-1157-z.

[34] P. Kindermann, F. Montecchiani, L. Schlipf, and A. Schulz. Drawing Subcubic 1-Planar Graphs with Few Bends, Few Slopes, and Large Angles. In T. Biedl and A. Kerren, editors, *Graph Drawing and Network Visualization*, pages 152–166. Springer, 2018. doi:10.1007/978-3-030-04414-5_11.

[35] G. W. Klau, K. Klein, and P. Mutzel. An experimental comparison of orthogonal compaction algorithms. In J. Marks, editor, *Graph Drawing*, pages 37–51. Springer, 2001. doi:10.1007/3-540-44541-2_5.

[36] J. Klawitter and J. Zink. Upward planar drawings with three and more slopes. In H. C. Purchase and I. Rutter, editors, *Graph Drawing and Network Visualization*, pages 149–165. Springer, 2021. doi:10.1007/978-3-030-92931-2_11.

[37] T. H. Kloepper and D. H. Huson. Drawing explicit phylogenetic networks and their integration into splitstree. *BMC Evolutionary Biology*, 8(1):22, 2008. doi:10.1186/1471-2148-8-22.

[38] K. Knauer, P. Micek, and B. Walczak. Outerplanar graph drawings with few slopes. *Computational Geometry*, 47(5):614–624, 2014. doi:doi.org/10.1016/j.comgeo.2014.01.003.

[39] W. Lenhart, G. Liotta, D. Mondal, and R. I. Nishat. Planar and Plane Slope Number of Partial 2-Trees. In S. Wismath and A. Wolff, editors, *Graph Drawing*, pages 412–423. Springer, 2013. doi:10.1007/978-3-319-03841-4_36.

[40] P. Mukkamala and D. Pálvölgyi. Drawing Cubic Graphs with the Four Basic Slopes. In M. van Kreveld and B. Speckmann, editors, *Graph Drawing*, pages 254–265. Springer, 2012. doi:10.1007/978-3-642-25878-7_25.

[41] M. Nöllenburg and A. Wolff. Drawing and labeling high-quality metro maps by mixed-integer programming. *IEEE Transactions on Visualization and Computer Graphics*, 17(5):626–641, 2011. doi:10.1109/TVCG.2010.81.

[42] J. Pach and D. Pálvölgyi. Bounded-degree graphs can have arbitrarily large slope numbers. *Electronic Journal of Combinatorics*, 13(1):N1, 2006. doi:10.37236/1139.

[43] A. Papakostas. Upward planarity testing of outerplanar dags. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing*, pages 298–306. Springer, 1995. doi:10.1007/3-540-58950-3_385.

[44] M. Patrignani. On the complexity of orthogonal compaction. *Computational Geometry*, 19(1):47–67, 2001. doi:10.1016/S0925-7721(01)00010-4.

[45] M. S. Rahman, M. Naznin, and T. Nishizeki. Orthogonal Drawings of Plane Graphs Without Bends. *Journal of Graph Algorithms and Applications*, 7(4):335–362, 2003. doi:10.7155/jgaa.00074.

[46] K. Schliep, M. Vidal-García, L. Biancani, F. H. Diaz, E. Ada, and C. Solís-Lemus. tanggle: Visualization of phylogenetic networks in a ggplot2 framework, 2021. URL: https://klausvigo.github.io/tanggle/articles/tanggle_vignette.html.

[47] A. Schulz. Drawing graphs with few arcs. *Journal of Graph Algorithms and Applications*, 19(1):393–412, 2015. doi:10.7155/jgaa.00366.

[48] M. Steel. *Phylogeny: Discrete and Random Processes in Evolution*. Society for Industrial and Applied Mathematics, 2016. doi:10.1137/1.9781611974485.

[49] R. Tamassia. On Embedding a Graph in the Grid with the Minimum Number of Bends. *SIAM Journal on Computing*, 16(3):421–444, 1987. doi:10.1137/0216030.

[50] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972. doi:10.1137/0201010.

[51] I. G. Tollis and K. G. Kakoulis. Algorithms for Visualizing Phylogenetic Networks. In Y. Hu and M. Nöllenburg, editors, *Graph Drawing and Network Visualization*, pages 183–195. Springer, 2016. doi:10.1007/978-3-319-50106-2_15.

[52] T. G. Vaughan. IcyTree: rapid browser-based visualization for phylogenetic trees and networks. *Bioinformatics*, 33(15):2392–2394, 2017. doi:10.1093/bioinformatics/btx155.

[53] G. A. Wade and J.-H. Chu. Drawability of Complete Graphs Using a Minimal Slope Set. *The Computer Journal*, 37(2):139–142, 1994. doi:10.1093/comjnl/37.2.139.